OPTIMAL AND ROBUST NEURAL NETWORK CONTROLLERS FOR PROXIMAL SPACECRAFT MANEUVERS

THESIS

Cole George, Capt, USAF

AFIT-ENY-MS-19-M-215

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, the United States Department of Defense or the United States Government. This is an academic work and should not be used to imply or infer actual mission capability or limitations.

AFIT-ENY-MS-19-M-215

OPTIMAL AND ROBUST NEURAL NETWORK CONTROLLERS FOR PROXIMAL SPACECRAFT MANEUVERS

THESIS

Presented to the Faculty
Department of Aeronautic and Astronautics
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Master of Science (Space Systems)

Cole George, BS
Capt, USAF

March 21, 2019

AFIT-ENY-MS-19-M-215

OPTIMAL AND ROBUST NEURAL NETWORK CONTROLLERS FOR PROXIMAL SPACECRAFT MANEUVERS

THESIS

Cole George, BS
Capt, USAF

Committee Membership:

Maj Joshuah A. Hess, PhD Chair	Date
Richard G. Cobb, PhD Member	Date
David W. Meyer, MS Member	Date

AFIT-ENY-MS-19-M-215

Abstract

Recent successes in machine learning research, buoyed by advances in computational power, have revitalized interest in neural networks and demonstrated their potential in solving complex controls problems. In this research, the reinforcement learning framework is combined with traditional direct shooting methods to generate optimal proximal spacecraft maneuvers. Open-loop and closed-loop feedback controllers, parameterized by multi-layer feed-forward artificial neural networks, are developed with evolutionary and gradient-based optimization algorithms. Utilizing Clohessy-Wiltshire relative motion dynamics, terminally constrained fixed-time, fuel-optimal trajectories are solved for intercept, rendezvous, and natural motion circumnavigation transfer maneuvers using three different thrust models: impulsive, finite, and continuous. In addition to optimality, the neurocontroller performance robustness to parametric uncertainty and bounded initial conditions is assessed. By bridging the gap between existing optimal and nonlinear control techniques, this research demonstrates that neurocontrollers offer a flexible and robust alternative approach to the solution of complex controls problems in the space domain and present a promising path forward to more capable, autonomous spacecraft.

The more you learn the less you know.

Acknowledgements

To my advisor, Maj Hess, for introducing me to the wonderful world of optimal control, for being approachable, for knowing when to offer guidance and when to step back, and for challenging me throughout. To Dr. Cobb for his endless enthusiasm and mentorship. To my professors, sponsors and classmates, their feedback, advice, and shared struggle are immeasurably appreciated. To my parents for instilling a tireless work ethic and for teaching me to take pride in every endeavor. To my sisters for raising me to be a better man. Lastly to Coda, who developed a Pavlovian response to the sound of my laptop shutting (indicating break time), wise beyond your years, you taught me patience. Thank you.

Cole George

Table of Contents

List of Figures

Figure Page

List of Tables

List of Abbreviations

CL	Closed-Loop
DU	Distance Units
DV	Decision/Design Variables
ECI	Earth-Centered Inertial
GEO	Geosynchronous
HCW	Hill-Clohessy-Wiltshire
ML	Machine Learning
NC	Neurocontroller
NLP	Nonlinear Program(ming)
NMC	Natural Motion Circumnavigation
OC	Optimal Control
OL	Open-Loop
RL	Reinforcement Learning
ROE	Relative Orbital Elements
RPO	Rendezvous & Proximity Operations
SL	Supervised Learning

List of Symbols

a	scalar
$\boldsymbol{a}$	vector
$\boldsymbol{A}$	matrix
$\lVert\cdot\rVert_2$	2-norm
t	time
$\dot{\boldsymbol{x}}$	time derivative
$\boldsymbol{r}$	position vector
$\dot{\boldsymbol{r}}$, $\boldsymbol{v}$	velocity vector
$\boldsymbol{x}$	state vector
$\boldsymbol{u}$	control vector
$\boldsymbol{X}$	m-by-n state history
$\boldsymbol{U}$	p-by-n control history
$\boldsymbol{\mathcal{N}}(\cdot)$	neural network function
$\boldsymbol{W}$	network weights
$\boldsymbol{b}$	network biases
$\boldsymbol{\theta}$	network parameters
T	thrust magnitude
$\boldsymbol{J}(\cdot)$	cost function
δ	variation
x^*	optimal value
σ	tolerance
γ	growth rate
n	mean motion
$\boldsymbol{\Phi}(t)$	state-transition matrix

a_e	relative semi-major axis
x_d	relative radial position offset
y_d	relative in-track position offset
β	relative phase angle
z_{max}	relative maximum cross-track position
ψ	relative orientation about radial axis

OPTIMAL AND ROBUST NEURAL NETWORK CONTROLLERS FOR
PROXIMAL SPACECRAFT MANEUVERS

I. Introduction

1.1 Motivation

The increasingly congested, contested, and competitive operational space environment is accelerating the need to examine responsiveness and survivability in this evolving domain, particularly in the realm of spacecraft rendezvous and proximity operations (RPO). Examples of RPO missions include the now common docking maneuvers performed by multiple space vehicles with the International Space Station over the last decade, and the Air Force's operational Geosynchronous (GEO) Space Situational Awareness Program for GEO-belt inspection.[1] Additionally, Air Force Research Laboratory RPO experiments with ANGELS and recently EAGLE and Mycroft further demonstrate that operating satellites effectively in close proximity is quickly becoming a vital guidance, navigation, and control task.[2, 3] The rising trend towards autonomy in other industries like robotics, transportation, computing, and manufacturing suggests a parallel pattern will emerge, if not already, in the space domain.[4] This growing demand for greater autonomy in space is amplified by challenges posed by the inherent nature of current satellite operations: intermittent control at a great distance. As well, a significant limiting factor for nearly all space vehicles is a finite supply of onboard fuel, and combined with a contested environment, controllers that are both optimal and robust form a requisite component for autonomous spacecraft operations. Concurrent with this autonomy trend, the field of machine learning (ML) has recently made significant progress in the solution of complex decision problems. Therefore, this research proposes that an opportunity exists to leverage the techniques advanced within ML towards the development of optimal and robust controllers essential to autonomous spacecraft RPO.

1.2 Research Overview

The hypothesis of this research is that neural networks offer a flexible parameterization that, through direct shooting and reinforcement learning methods, can be leveraged to develop optimal and robust controllers for terminally constrained, fixed-time minimum control proximal spacecraft maneuvers using evolutionary and gradient-based optimization techniques. The resulting neural networks are referred to as neurocontrollers.

1.2.1 Questions.

To address the hypothesis, the following research questions are posed:

1. How can neural networks be used to parameterize an open- and closed-loop control for impulsive, finite, and continuous thrust models and solved for common proximal spacecraft maneuvers?

2. How can neural networks be used to produce a single open-loop controller capable of generating optimal control guidance for a predefined bounded set of initial states?

3. How can neural networks be used to produce a single closed-loop controller capable of generating optimal control guidance for a predefined bounded parametric uncertainty in the state dynamics?

1.2.2 Tasks.

The following tasks must be performed in this research to answer the corresponding research questions:

1. Use a direct shooting method to optimize a control parameterized by a neural network for combinations of open- and closed-loop controllers, thrust models, and common proximal spacecraft maneuvers.

2. Combine reinforcement learning techniques with direct shooting methods to optimize an open-loop control parameterized by a neural network valid for a range of initial states using a heuristic optimization algorithm.

3. Combine reinforcement learning techniques with direct shooting methods to optimize a closed-loop control parameterized by a neural network robust to bounded parametric uncertainty in the state dynamics using a heuristic optimization algorithm.

4. Validate neurocontroller performance for the problems investigated by comparing to existing results in literature or develop optimal benchmark solutions.

1.2.3 Scope.

This work is based solely on simulation with no experimental investigation. The dynamics models used are entirely deterministic. The only stochastic elements are the random components inherent to heuristic optimization, initial guess seeding, and batch sampling. Numerical optimization methods are employed that do not guarantee global extrema, but in the case of gradient-descent methods, may ensure local minima. Outer-loop controllers are solved off-line and assume an inner-loop controller exists onboard the spacecraft that is capable of executing the generated control profile. Lastly, the controls are solved exclusively for fixed-time maneuvers.

1.3 Assumptions and Limitations

The spacecraft maneuver problems investigated in this research assume relative motion in near-circular orbits. The relative orbits and maneuvers examined are valid only for distances in close proximity (i.e., on the order of 100 km at geosynchronous orbit) to the specified reference origin and scales with the altitude of the origin. Impulsive, finite, and continuous thrust models are implemented, with thrust taking the form of simple directed acceleration vectors and mass loss neglected; this represents a reasonable approximation for the duration (e.g. less than half an orbit) and size of maneuvers investigated. Additionally, no maximum slew rate is imposed allowing the spacecraft to change acceleration direction instantaneously. The instantaneous magnitude of control is typically unbounded with the exception of some finite control examples. Throughout this work full state information is assumed to be available with no uncertainty.

Overview.

The remainder of this document is divided into four chapters. Chapter 2 provides a background of topics pertinent to this research and reviews relevant literature. Chapter 3 details the methodology employed within this work. Chapter 4 implements the methodology and presents an analysis of the results. Finally, Chapter 5 concludes the thesis with a discussion of key insights and future work.

II. Background

2.1 Overview

This research broadly concerns three primary subject areas: (a) relative satellite motion, (b) controls and optimization, and (c) machine learning. This chapter devotes a background section to each topic with discussion limited to relevant sub-topics with pertinent literature highlighted. In particular, as machine learning encompasses a vast field of study, emphasis is placed on the reinforcement learning sub-field and the concept of neural networks especially. The three main problems explored in this research are then summarized in the concluding section.

2.2 Relative Satellite Motion

When studying the motion of multiple satellites in close proximity, or merely a single satellite in its local region, it is convenient to work in a relative frame. In this context, the definition of *proximity* depends on the employed dynamics model as well as the altitude and time period of interest. Figure 1 depicts the relative frame used throughout this work, where $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$ represent the relative Hill frame components in terms of the $\boldsymbol{ijk}$ Earth-centered inertial (ECI) frame. The origin of this relative frame is commonly referred to as the chief (or target) and any other satellites as deputies (or interceptors). In this frame, $\boldsymbol{x}$ is co-linear with the position vector, $\boldsymbol{y}$ is in the direction of the velocity vector aligned with the local horizontal, and $\boldsymbol{z}$ is normal to the orbit plane. The xyz notation is consistent with the RSW coordinate system as defined in [5]. For this work, $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$ are equivalent to the radial, in-track, and cross-track components respectively.

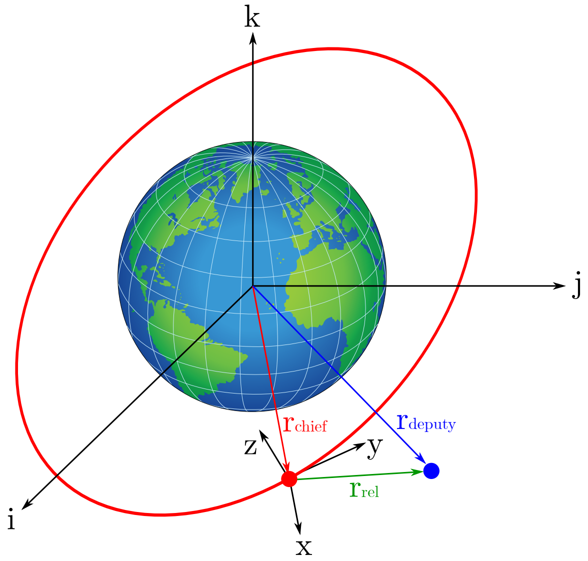


Figure 1. Relative Hill Frame

2.2.1 Dynamics Models.

Several dynamics models have been developed over the past half-century.[6, 7] The Hill-Clohessy-Wiltshire (HCW) model is a linear approximation for the relative motion of spacecraft in close-proximity on near-circular orbits.[8] This is the model used throughout this research and is discussed in more detail in § 2.2.2. For near-circular orbits, a second-order solution was developed, often referred to as Quadratic-Volterra, that increases the valid separation distance.[9, 10] Butcher developed higher-order solutions for circular and slightly-eccentric orbits by incorporating the effects of eccentricity as perturbation resulting in a system explicit in time.[11] Tschauner and Hempel developed a linear time-varying solution by normalizing the coordinates and changing the independent variable from time to true anomaly that improved the accuracy for slightly eccentric orbits compared to HCW, but suffered from singularities at zero eccentricity.[12, 13] Overcoming these limitations, Yamanaka and Ankersen developed a nonsingular, linear solution for arbitrarily eccentric orbits.[14] Further, Willis, Lovell and D'Amico introduced a second-order translational-state solution for arbitrarily eccentric orbits.[6] Additional models exist that account for perturbations, such as Schweighart and Sedwick for the J2 effect, and the Gim-Alfriend state transition matrix that accounts for J2 and eccentricity.[15, 16]

2.2.2 Hill-Clohessy-Wiltshire Equations.

For this research, the linearized Hill-Clohessy-Wiltshire equations of relative satellite motion are used exclusively as they offer a convenient and computationally efficient set of dynamics while providing a sufficient level of accuracy for the domain of study. The methodology developed however is such that higher-order dynamics may be substituted with relative ease, albeit likely at the cost of computation time.

Unforced and Forced Motion.

The HCW differential equations are given in Eq. (2.1).[5, 8] These unforced equations of motion assume no perturbations and the origin, or chief, is in a circular orbit.

$$\begin{aligned} \ddot{x} &= 3n^2 x + 2n\dot{y} \\ \ddot{y} &= -2n\dot{x} \\ \ddot{z} &= -n^2 z \end{aligned} \tag{2.1}$$

where x, y, and z represent the radial, in-track, and cross-track components respectively, and n the mean motion,

$$n = \sqrt{\frac{\mu}{a^3}} \tag{2.2}$$

such that μ is the standard gravitational parameter ($\approx 3.986 \cdot 10^5\,\mathrm{km}^3/\mathrm{s}^2$ for Earth) and a is the semi-major axis of the specified origin. Conveniently in this formulation the z-component is decoupled from the x and y dynamics. Nonzero values of z are considered out-of-plane motion. Throughout this research, all three dimensions are employed, although some scenarios are chosen to be planar. These unforced equations of motion imply there are no external forces on the interceptor and any

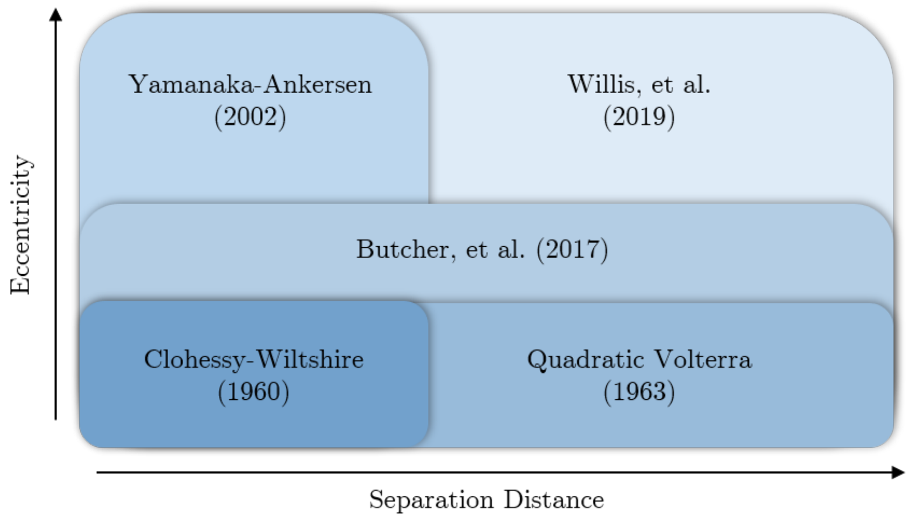


Figure 2. Overview of Relative Satellite Motion Models [6]

$\Delta V's$ are impulsive such that the resulting velocity becomes the initial condition for the next state propagation.[5] These equations may be adapted for forced motion by appending simple accelerations to the dynamics as shown in Eq. (2.3), and may be solved through numerical integration, convolution, or analytically. For this research, the controls solved for are directly these accelerations, with mass loss neglected, and numerically integrated.

$$\begin{aligned}\ddot{x} &= 3n^2x + 2n\dot{y} + a_x \\ \ddot{y} &= -2n\dot{x} + a_y \\ \ddot{z} &= -n^2z + a_z\end{aligned} \tag{2.3}$$

For unforced motion, a closed-form solution to Eq. (2.1) exists, given in Eq. (2.4).

$$\boldsymbol{x}(t) = \boldsymbol{\Phi}(t)\boldsymbol{x}(t_0),\ \boldsymbol{x} = [x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}]^T$$

where

$$\mathbf{\Phi}(t) = \begin{bmatrix} 4-3\cos nt & 0 & 0 & \frac{1}{n}\sin nt & \frac{2}{n}(1-\cos nt) & 0 \\ 6(\sin nt - nt) & 1 & 0 & \frac{2}{n}(\cos nt - 1) & \frac{1}{n}(4\sin nt - 3nt) & 0 \\ 0 & 0 & \cos nt & 0 & 0 & \frac{1}{n}\sin nt \\ 3n\sin nt & 0 & 0 & \cos nt & 2\sin nt & 0 \\ 6n(\cos nt - 1) & 0 & 0 & -2\sin nt & 4\cos nt - 3 & 0 \\ 0 & 0 & -n\sin nt & 0 & 0 & \cos nt \end{bmatrix} \tag{2.4}$$

This state transition matrix can be used to efficiently propagate the equations of unforced motion.

Assumptions.

These equations assume the motion of the deputy spacecraft remains in close proximity to the chief. This error scales with the size of the chief's semi-major axis, such that a separation distance of 100 km at geosynchronous orbit is a more reasonable operating range than at low-Earth orbit. For this research, the mean motion is taken to be at geosynchronous orbit, albeit the distance units used are dimensionless, this proximity requirement should be kept in mind when translating to actual distance units. Additionally, the approximation error grows with time and consequently the duration of all trajectories examined in this research are less than one orbit. Finally, these equations assume the chief is in a circular orbit, or in practice a near-circular orbit, and no perturbations exist, although variations of these equations have been developed to accommodate elliptical and perturbed orbits as discussed in § 2.2.1. For the proof of concept studies examined in this research, these assumptions represent a reasonable approximation without overly-restricting the general conclusions that may be drawn from the results obtained or methods developed.

2.2.3 Relative Orbital Elements.

It is convenient when working with spacecraft proximity operations to employ a coordinate transformation that allows a more intuitive specification of relative orbital geometry, as developed in [17]. The relative orbital elements (ROE) are depicted in Figure 3, where a_e is the semi-major axis of a 2-by-1 ellipse, x_d and y_d are the origin offset, β is the in-plane phasing angle. Not pictured is z_{max}, the maximum cross-track component, and ψ, the orientation about the z-axis. Note that Figure 3 depicts the instantaneous geometry of the deputy and that a drift component may exist. This transformation can be readily computed from the cartesian representation in the HCW frame to the ROE's,

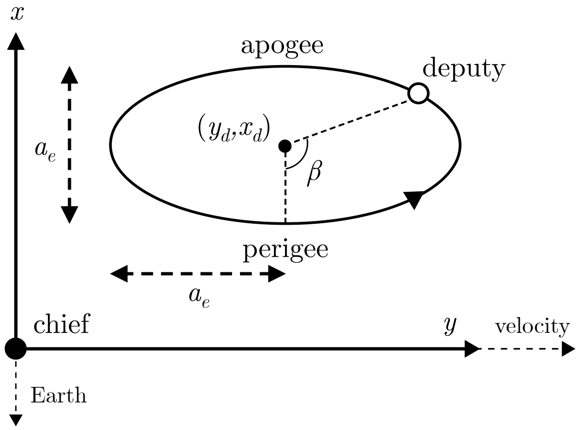


Figure 3. Relative Orbital Elements [17]

$$
\begin{aligned}
a_e &= 2\sqrt{\left(\frac{\dot{x}}{n}\right)^2 + \left(3x + 2\frac{\dot{y}}{n}\right)^2} \qquad & \beta &= \operatorname{atan2}(\dot{x}, 3nx + 2\dot{y}) \\
x_d &= 4x + 2\frac{\dot{y}}{n} & z_{max} &= \sqrt{\left(\frac{\dot{z}}{n}\right)^2 + z^2} \\
y_d &= y - 2\frac{\dot{x}}{n} & \psi &= \operatorname{atan2}(nz, \dot{z})
\end{aligned}
\tag{2.5}
$$

and the reverse,

$$\begin{aligned} x &= -\frac{a_e}{2}\cos\beta + x_d & \dot{x} &= \frac{a_e}{2}n\sin\beta \\ y &= a_e\sin\beta + y_d & \dot{y} &= a_e n\cos\beta - \tfrac{3}{2}nx_d \\ z &= z_{max}\sin\psi & \dot{z} &= z_{max}n\cos\psi \end{aligned} \tag{2.6}$$

A special case exists when $x_d = 0$ resulting in a so-called natural motion circumnavigation (NMC) whereby the unforced motion of the deputy spacecraft traces out a closed 2-by-1 ellipse relative to the chief. The utility of this orbit is discussed further in § 2.2.4. Additionally, the ROE's may be propagated directly as functions of time, and are equivalent to propagating the unforced HCW equations.

$$\begin{aligned} a_e &= a_{e0} & \beta &= \beta_0 + nt \\ x_d &= x_{d0} & z_{max} &= z_{max0} \\ y_d &= y_{d0} - \frac{3}{2}nx_d t & \psi &= \psi_0 + nt \end{aligned} \tag{2.7}$$

For convenience, all dynamics in this research are propagated using the HCW equations, and appropriate transformations made only at the initial and final states.

2.2.4 Rendezvous and Proximity Operations.

The term Rendezvous and Proximity Operations (RPO), as the name suggests, refers to spacecraft operations conducted in close proximity (see § 2.2.2) to itself or another satellite(s). With regards to RPO, a few common trajectories and relative orbits are introduced and implemented in this research, summarized in Table 3, where $\boldsymbol{r}$ is the relative position vector. The leader-follower and teardrop orbits are not implemented in this research, but are included here for completeness as the methodology presented may be readily applied to these orbits as well.

Intercept & Rendezvous.

In this context, intercept is taken to mean a spacecraft must match position only for a designated target. This implies that velocity is not constrained. Rendezvous however is taken to mean a spacecraft must match the position and velocity for a designated target. Docking with another satellite is an example of rendezvous and was the original motivation for the development of the HCW equations.[8]

Natural Motion Circumnavigation.

When $x_d = 0$, the spacecraft traces out a closed 2-by-1 ellipse in the relative frame. This allows the deputy to circumnavigate the chief spacecraft indefinitely without using fuel. The NMC has a period equal to the period of the chief's orbit, with the size of the ellipse determined by a_e. This type of relative orbit may be useful for inspection missions and staging prior to docking.

Leader-Follower.

A leader-follower relative orbit is such that only an in-track component exists, i.e. both satellites are in the same orbit with a phase difference. Equivalently, this can be thought of as a special case of the NMC requirements where $a_e = 0$. A leader or follower orbit is specified by setting $y_d > 0$ or $y_d < 0$ respectively.

Teardrop.

The teardrop, or pogo, trajectory is a useful relative orbit whereby a quasi-hovering motion results in the deputy spacecraft and requires $a_e > \frac{3}{2}|x_d|$. Once again, this is useful for inspection missions where it is desired to stay within a specified angle of the chief for a period of time. A number of parameters are developed in [18] to specify the geometry of the teardrop, including the location, width, closest approach, and teardrop period. The teardrop may be repeated indefinitely with minimal fuel by executing a burn at the cusp.

Table 3. Summary of Common Relative Maneuvers & Orbits

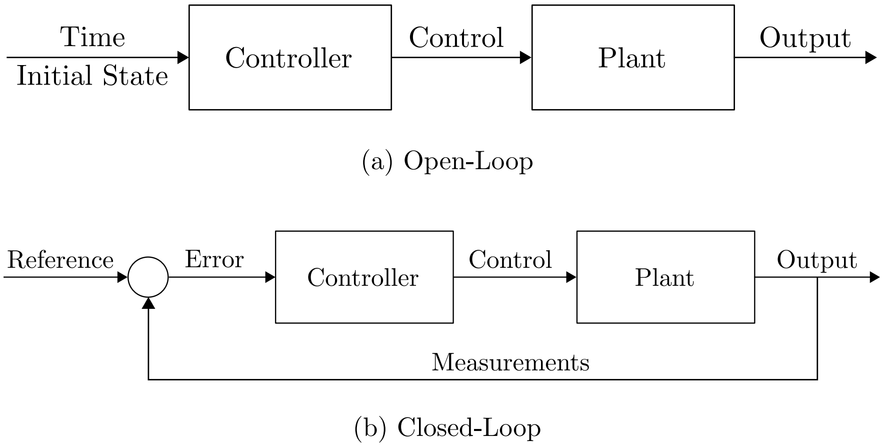
Intercept	$\boldsymbol{r}(t_f) = \boldsymbol{r}_{des}$
Rendezvous	$\boldsymbol{r}(t_f) = \boldsymbol{r}_{des},\ \dot{\boldsymbol{r}}(t_f) = \dot{\boldsymbol{r}}_{des}$
NMC	$x_d = 0,\ a_e \neq 0$
Leader-Follower	$x_d = a_e = 0$
Teardrop	$a_e > \frac{3}{2}\lvert x_d \rvert$

2.3 Control and Optimization

This section introduces some foundational control concepts and the topic of optimal control along with classic solution approaches.

2.3.1 Control Loops and Thrust Models.

A controller may be implemented in either an open- or closed-loop configuration.[19] For this research, an open-loop (OL) controller (Fig. 4a) is one that does not depend on future states and consequently the control history for the entirety of the trajectory may be determined independent of propagation. An OL controller may depend on the initial state, any other fixed parameter, or time. Open-loop is the primary control of interest to the optimal control community. In contrast, closed-loop (CL) controllers (Fig. 4b) incorporate a feedback element, and for this research are explicitly dependent on some representation of the current state, and consequently the control can not be determined prior to propagation. CL controllers are required to handle uncertainties in the environment or system dynamics. Although in practice feedback typically implies measurements and estimation, for this research certain full-state (or subset thereof) feedback is assumed. For a given problem, the OL and CL trajectories are equivalent. Figure 10 and subsequent discussion provide further distinction between OL and CL in the context of neural network controllers. For this research, three different thrust models are employed: impulsive, finite, and continuous. For each, mass loss is neglected.



(a) Open-Loop

(b) Closed-Loop

Figure 4. Notional Open- and Closed-Loop Control Block Diagrams

Impulsive Thrust.

Impulsive thrust assumes an instantaneous change in velocity, and as the name implies is often referred to as ΔV. Although not physically realizable, the impulsive model is a reasonable approximation when the duration of the thrust(s) is small relative to the overall trajectory, especially if the thrust direction remains relatively constant throughout the burn. With regards to the dynamics, impulsive thrusts are incorporated by using the resultant velocity vector, i.e. the current velocity plus the ΔV, as the initial conditions for an unforced motion segment found by either numerically integrating or using the state transition matrix. With this model, the variables of interest are magnitude and direction.

Continuous Thrust.

Continuous thrust models refer to the continuous application of controlled acceleration throughout a given trajectory. These trajectories are numerically integrated using the forced HCW equations of motion. Continuous thrust provides a good model for the electric propulsion systems that are seeing increased usage aboard spacecraft and the mathematical derivation of optimal control laws traditionally assumes continuous control (see § 2.3.2). As with impulsive thrust, the variables of interest are magnitude and direction.

Finite Thrust.

Finite thrust represents a more physically realizable control model that is somewhere between impulsive and continuous. In this context, finite thrust refers to a trajectory composed of continuous control acceleration segments separated by unforced motion segments. During the controlled segments, the control may be defined by constant or variable throttle, direction and duration, summarized in Table 4. In each case the values may be fixed *a priori* or as variables to be optimized. The controlled segments are once again numerically integrated using the forced HCW equations of motion while the uncontrolled segments are propagated with either the state transition matrix or numerical integration. Although in actual spacecraft the thrust magnitude may only take discrete values, for this research continuous thrust values are assumed. Therefore, depending on the combination used, the variables of interest are magnitude (throttle), direction, and duration of burn.

2.3.2 Optimal Control.

Optimal control (OC) encompasses a diverse field of study at the intersection of optimization and control theory with its foundations in the calculus of variations. [20,

Table 4. Finite Thrust Control Combinations

Throttle	Direction	Duration
Constant	Constant	Variable
Constant	Variable	Variable/Constant
Variable	Constant	Variable/Constant
Variable	Variable	Variable/Constant

21] At its most basic, OC is concerned with finding an admissible control history that minimizes a performance index subject to some dynamic constraints (e.g., differential equations of motion, path constraints, and/or boundary conditions). In general, the goal of OC is to determine $\boldsymbol{x}^*(t)$ and $\boldsymbol{u}^*(t)$ that minimizes the cost functional (i.e. a mapping from a vector space to a scalar)

$$J = \Phi\big(\boldsymbol{x}(t), t_f\big) + \int_{t_0}^{t_f} \mathcal{L}\big(\boldsymbol{x}(t), \boldsymbol{u}(t), t\big)\, dt \tag{2.8}$$

where Φ is the terminal cost and $\mathcal{L}$ is the running cost, subject to the dynamic constraint

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) \tag{2.9}$$

the boundary conditions

$$\phi\big(\boldsymbol{x}(t_0), t_0, \boldsymbol{x}(t_f), t_f\big) = 0 \tag{2.10}$$

and path constraints

$$\boldsymbol{C}_{min} \leq \boldsymbol{C}\big(\boldsymbol{x}(t), \boldsymbol{u}(t)\big) \leq \boldsymbol{C}_{max}. \tag{2.11}$$

Direct and Indirect Methods.

Two methods are widely used to solve OC problems: direct and indirect optimization methods.[22] Indirect methods explicitly solve the optimality conditions while direct methods do not require this analytic expression and instead introduce a parametric representation to form a nonlinear programming (NLP) problem. While indirect methods can guarantee local optimality if the necessary and sufficient conditions are met, complex OC problems can often prove to be intractable. Direct methods are the focus of this research. Several implementations of the direct method exists, primarily differentiated by the parametric representation. In this context, single- and multiple-shooting methods refer to control parameterization in a single or multiple trajectory segments respectively. For example, the control law may be assumed to be linear with respect to time, such that

$$u(t) = p_2 t + p_1 \tag{2.12}$$

where p_i are the parameters to be found, thereby permitting the controlled trajectory to be integrated directly, either analytically or numerically. In addition to the control, collocation methods parameterize the state as well, thereby approximating the dynamics and forming an NLP. Several collocation methods exist with increasing order of accuracy, from trapezoidal to Hermite-Simpson.[23] Pseudospectral methods are an extension of collocation methods that employ Gaussian quadrature points to approximate the function integral and Lagrange interpolating polynomials to approximate the differential equations.[24] Once more, this research exclusively uses a direct single-shooting method. Regardless of the method employed to transform the optimal control problem, the result is typically an NLP, that may be solved through a variety of gradient-based and/or heuristic methods.

Nonlinear and Robust Control.

A shortcoming of the OC methods presented here is that their solution is only applicable to the single set of values for which the problem scenario is solved. For example, if the initial position changes, the entire control problem needs to be solved again. Nonlinear control techniques such as feedback linearization, sliding mode and adaptive control methods, on the other hand can be used to create a controller that is robust to a variety of conditions.[25] These conditions can be anything from initial or final states, uncertain parameters in the dynamics, etc. Depending on the employed technique, the controller may be shown to be stable (in some sense, e.g. Lyapunov stable) for only a bounded range of uncertainty. However, these methods do not typically result in an optimal control as defined previously. Model-predictive control is one method that attempts to address this trade-off by repeatedly solving on-line an OC problem for a specified horizon. One of the shortcomings of this approach is that performing the optimization requires capable onboard hardware. For this research, a controller that produces a control that is optimal with respect to a specified performance index throughout a specified bounded operating range, is considered to be both optimal and robust.

2.3.3 Nonlinear Programming.

The direct shooting method employed in this research transforms the optimal control problem into a nonlinear program. A nonlinear program is a general mathematical model whereby a cost function is minimized while satisfying all nonlinear equality and inequality constraints.[26] Within this model, all equality constraints are set to zero and inequality constraints transformed to be less than or equal to zero. The standard NLP formulation is presented below:

$$\begin{aligned} &\underset{\boldsymbol{x}}{\text{minimize}}\; f(\boldsymbol{x}) \\ \text{s.t.}\quad & h_j(\boldsymbol{x}) = 0,\ \forall\, j = 1 \ldots p \\ & g_i(\boldsymbol{x}) \leq 0,\ \forall\, i = 1 \ldots m \end{aligned} \tag{2.13}$$

Note that once the OC problem has been transcribed into an NLP, the cost function is no longer a cost functional. Mature and powerful software tools exist to solve

NLP problems, albeit not as robust as for linear programs. Two different optimization methods are used within this research to solve the NLP's: gradient-descent and heuristic. Several gradient-descent algorithms exist to solve constrained NLP's such as sequential quadratic programming that rely on the gradient of the cost function and constraints as well as second-order information to find local minima. Heuristic methods however do not generally rely on differentiation and instead use stochastic techniques to search the design space. The heuristic method used within this research is a genetic algorithm, that takes inspiration from genetic evolution, and evolves a population of candidate solutions.

2.4 Machine Learning and Neural Networks

This section provides an overview of machine learning with a focus on reinforcement learning as applied to traditional controls problems. Relevant research is discussed, particularly as it pertains to the field of spacecraft control.

2.4.1 Learning Types.

Machine learning (ML) is a family of statistical techniques used to approximate data relationships without explicit instruction, most commonly implemented via artificial neural networks (ANN), with myriad applications in image recognition, data analytics, forecasting, fault detection, and robotics.[27] Although the concept of ML has existed for over half a century [28], the past two decades have seen a surge in ML-related research [29] particularly with the advent of deep learning (deep simply refers to the multiple hidden layers within a network, see § 2.4.2), due at least in part to the rapid rise in computational power over that same period. Consequently, some aspects of the field remain very much in their infancy. Combine this with the production and application of custom built processors that excel at the kinds of large-scale matrix calculations inherent to neural networks, and this suggests the upward trend is likely to continue in the near future.

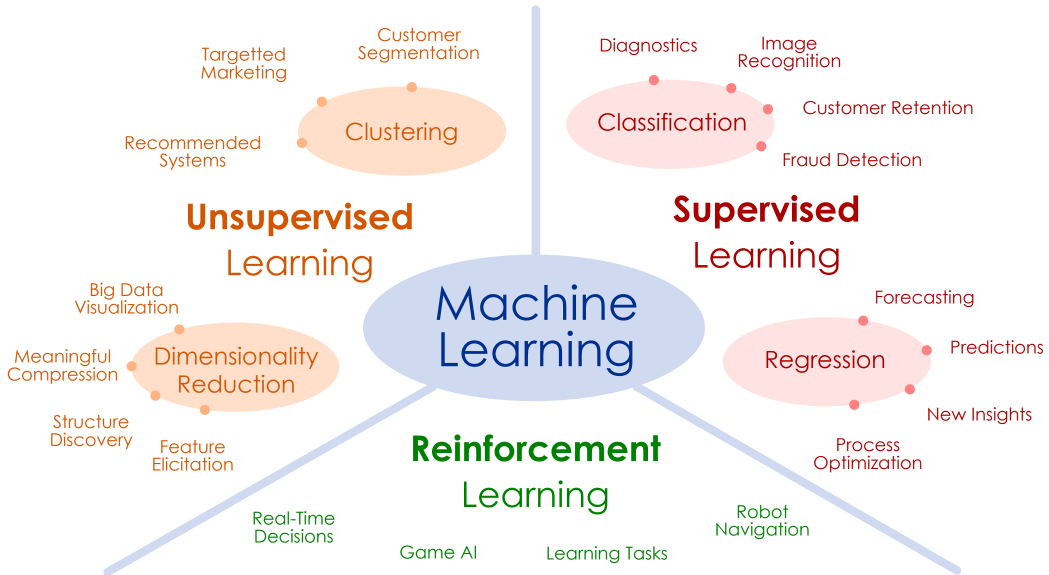


Figure 5. Branches of Machine Learning [30]

The umbrella of machine learning can be broadly divided into three major categories: supervised, unsupervised, and reinforcement learning.[27] Supervised learning (SL) attempts to find a relationship, or mapping, between labeled input-output data while unsupervised learning attempts to extract patterns in unlabeled data. The implementation of the exclusive-or (XOR) function, shown in Table 5, is a simple example of SL where the mapping between two binary inputs and a single binary output must be learned.

Table 5. XOR Truth Table

Input 1	Input 2	Output
0	0	0
0	1	1
1	0	1
1	1	0

In this case, because the true output is known, the optimization algorithm seeks to minimize the difference, e.g. mean-square error (MSE), between the network's output $\hat{y}$ and truth y by varying the parameters $\boldsymbol{\theta}$ that define the network, as in Eq. (2.14).

$$\min_{\boldsymbol{\theta}} J = \sum_{i=1}^{4} \|y - \hat{y}\|_2^2 \tag{2.14}$$

Although this example is limited to only four possible input combinations, both supervised and unsupervised methods typically require large datasets that, depending on context, are prohibitively difficult to obtain or generate. In contrast, the third category of ML, reinforcement learning (RL), does not require a large *a priori* labeled dataset, and instead attempts to discover an optimal policy directly through the interaction of an agent in a specified environment according to some user-defined reward function. In other words, the desired output may not be known precisely, but some performance index or desirable properties are known. The mountain car problem depicted in Figure 6 is a benchmark RL problem wherein a car starts from a random position near the bottom of a valley with the goal to reach the top of the hill. However, the car does not have the necessary power to go directly up the hill and therefore must *learn* to go back first and then leverage its momentum.[31, 32] Variations of this problem exist with different performance indices such as minimizing

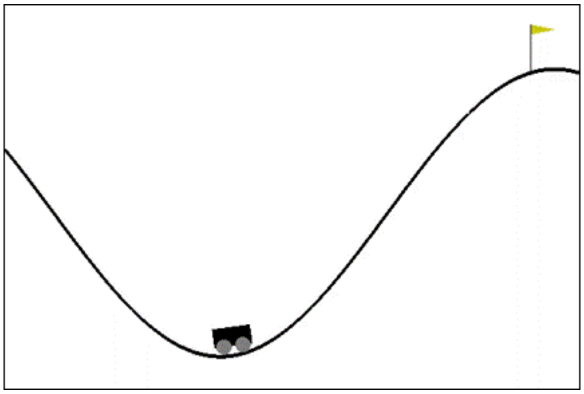


Figure 6. Mountain Car Problem [31]

control or time. From this example the similarity between RL and traditional optimal control is readily observed. Figure 7 depicts a typical closed-loop controller block diagram with the equivalent terminology identified. RL is the primary inspiration of this research; aspects of RL are combined with traditional optimal control methods to develop and implement the RL-like approach presented here.

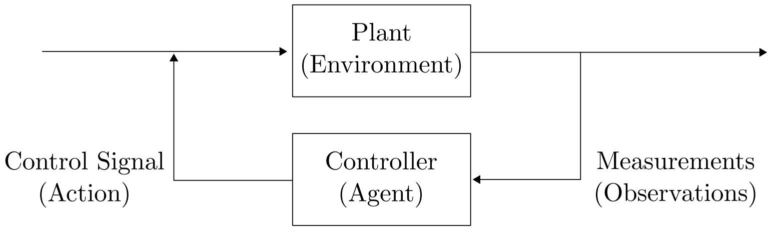


Figure 7. RL and Classical Controls Terminology and Block Diagram

In [33–36] RL is used to solve large, discrete state-action space games, such as backgammon, chess, go, and Atari. The latter two problems in particular demonstrated the efficacy of deep RL to existing RL methods like Q-learning. When applied to high-dimensional dynamic problems with continuous state-action spaces (states and controls may take on any continuous value) however, these methods often become intractable. To overcome this, actor-critic Q-learning variations were developed.[37] A software framework for testing RL algorithms on a standardized library of continuous and discrete environments like Atari, robotics, and classical control problems was developed in [31]. [38] presents several implementations of a class of RL control methods that bridge the gap between optimal and adaptive control called approximate dynamic programming (ADP) using Q-learning and integral RL to solve the Hamilton-Jacobi-Bellman (HJB) equation online. All of these methods use gradient-based optimization to solve for the state-action value function and provide feedback at each step. Recently however, [39] demonstrated evolutionary strategies (ES) as a feasible alternative approach, using heuristic evolutionary optimization techniques like genetic algorithms, differential evolution, and co-variance matrix adaptation to solve for the optimal network parameters without attempting to approximate the value function. In other words, traditional RL rewards actions while ES rewards policies. ES exhibits strengths in environments with sparse rewards and long time horizons and is indifferent to continuous or discrete state-action spaces. While there is some debate regarding the distinction between RL and ES, this research considers ES a subset of RL, and regardless the optimization method used, i.e. gradient-based or heuristic, the policy rewarding approach is exclusively used.

2.4.2 Neural Networks.

Neural networks are increasingly the standard mechanism by which ML is employed. Neural networks, sometimes called *artificial* neural networks to distinguish them from their biological source of inspiration, are simply nonlinear matrix functions that can be used to approximate an arbitrary input-output mapping.[27] ANN's typically consist of an input layer, one or more hidden layers, and an output layer, see Figure 8. Each layer comprises multiple neurons, or nodes, each with an associated element-wise nonlinear operator, or activation function. Some common activation functions are depicted in Figure 9. Although a variety of neural network architectures

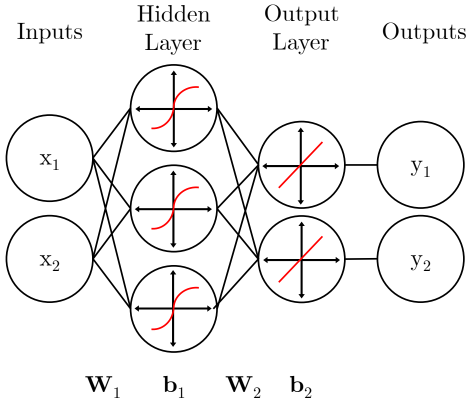
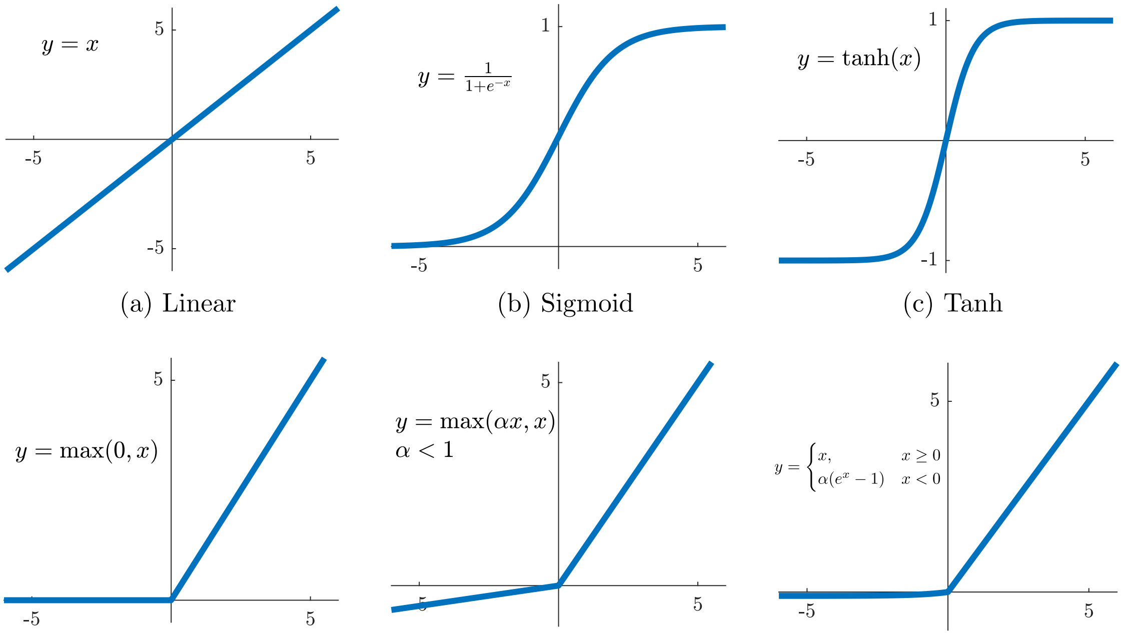


Figure 8. Notional Feed-Forward Neural Network

exist, from recurrent neural networks to convolutional neural networks (used especially in image recognition), this research exclusively uses multi-layer feed-forward networks whereby all nodes of a layer are connected to all nodes of the next layer. A useful property of these networks is their ability to approximate a wide class of functions, as shown in [41], thus earning the nickname *universal approximators.* The number of nodes in each layer is often referred to as the *width* of the network and the number of layers as the *depth* of the network, roughly corresponding to the representational capacity of the network and its accuracy respectively. The required width and depth depend on the specific problem to be solved (for example, the XOR example requires at least one hidden layer), but wider and deeper networks drastically increase the number of parameters to be tuned thereby increasing computation time–there is always a trade-off. In addition to the architecture, or topology of the network, the primary parameters for such networks are the weights of each node connection and



(a) Linear

(b) Sigmoid

(c) Tanh

(d) Rectified Linear Unit (ReLU)

(e) Leaky ReLU

(f) Exponential Linear Unit (ELU)

Figure 9. Common Activation Functions [40]

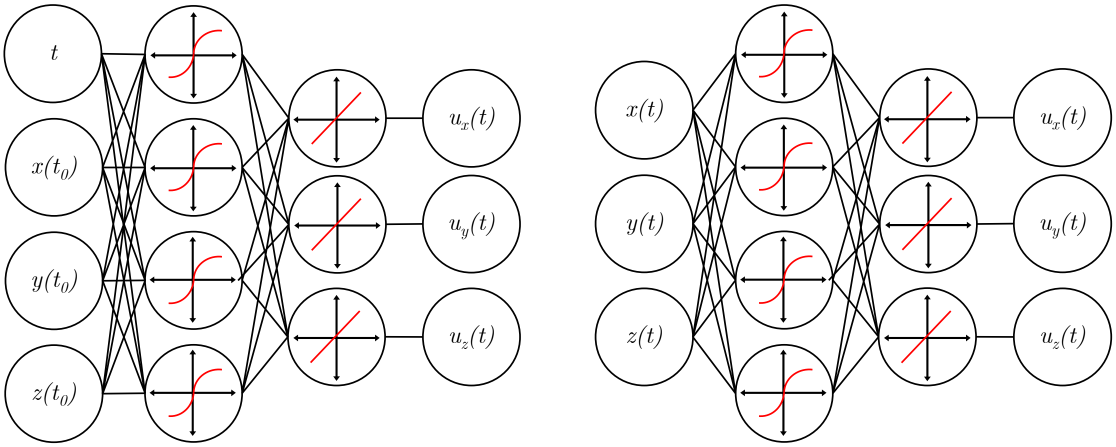
the node biases, collectively forming the parameter vector, $\boldsymbol{\theta}$. While these parameters are often optimized with stochastic gradient descent techniques through backpropagation, other optimization methods may be used to include heuristic techniques. As in Eq. (2.15), in general the goal is to find the mapping $\mathbf{N}(\boldsymbol{\theta})$ from some input vector, or scalar, $\boldsymbol{x}$ to the optimal output vector, or scalar, $\boldsymbol{y}$ that minimize some cost, or loss, function. In this context $\boldsymbol{x}$ may or may not correspond to the state vector $\boldsymbol{x}$ in Eq. (2.4).

$$\mathbf{N}(\boldsymbol{\theta}):\ \boldsymbol{x} \rightarrow \boldsymbol{y} \tag{2.15}$$

2.4.3 Spacecraft Control with Neural Networks.

While the majority of RL research to date has largely focused on discrete games, the recent advances being made in ML and with neural networks in general has inspired a gradual reassessment of their application to other domains, specifically the classical controls field. Further, several examples exist wherein RL and SL approaches have been taken to solve spacecraft controls problems. For low-thrust interplanetary missions, [42] used an evolutionary RL approach to develop a closed-loop evolutionary neurocontroller (NC) comprising a three-layer feed-forward network and was the initial inspiration for this work. Izzo solved similar interplanetary low-thrust problems using SL techniques instead.[43–46] Specifically [43] uses evolutionary optimization within SL approach for an Earth-Mars orbital transfer requiring the generation of a large dataset using a traditional optimal control technique, in this case solving over 300,000 two-point boundary value problems. This dataset is then used to train the neural network thereby providing a control interpolating function. Horn et al. used neural networks to generate optimal trajectories for unmanned aerial vehicles as an alternative to collocation and pseudospectral methods.[47] In [48], evolutionary methods are used to optimize neural networks in an RL manner to generate optimal steering control gains for low-thrust spiral trajectories. Much work has also been accomplished using neural networks for on-board fault detection and the solution of spacecraft attitude control problems, specifically Dracopoulos's extensive work with evolutionary neural network adaptive control.[49–51] Within the domain of relative spacecraft control, Youmans and Lutze used SL and Clohessy-Wiltshire dynamics to develop optimal open- and closed-loop neural network controllers for spacecraft.[52]

This research is most closely related to Youmans but uses an RL-like approach and extends the application to a variety of RPO maneuvers and control models. This approach bypasses the initial dataset generation task required of SL and solves the problem of spacecraft proximity control directly, similar to RL but with explicitly defined nonlinear terminal constraints incorporated into the optimization. Additionally, while much of the traditional ML controls research to date focuses on closed-loop feedback control, open-loop controllers are also developed within this work. Figure 10 depicts notional open- and closed-loop neural network controllers: open-loop depends on the initial state and possibly the changing time value, while closed-loop incorporates some sort of feedback in this case the current state (see § 2.3.1 for further discussion of open- and closed-loop control).



(a) Open-Loop

(b) Closed-Loop

Figure 10. Notional Open- and Closed-Loop Neural Network Controllers

2.5 Problems Overview

This research is divided into three major problems: A, B, and C. Problem A is further subdivided into three sub-problems: A1, A2, and A3. Problems B and C build on the concepts developed in Problem A. Together, the five problems are primarily distinguished by the employed control configuration (i.e. open- or closed-loop), control type (i.e. impulsive, finite, or continuous), and scenario to be solved (i.e. intercept, rendezvous, or NMC transfer). These differences illustrate the flexible application of neural network controllers. Additionally, Problems B and C differ from Problem A by demonstrating a certain level of robustness.

Problem A: Optimal Neural Network Controllers.

The three sub-problems within Problem A use a different combination of control configuration and type to solve each an intercept, rendezvous, and NMC transfer scenario. The resulting controllers are only optimal and valid for the precise problem parameters specified and are thus not considered robust in this context.

Problem B: Bounded Initial Conditions.

Problem B implements an optimal impulsive open-loop controller in an intercept scenario, building on concepts developed in Problems A1 and A2. This problem introduces the idea of bounded initial conditions, resulting in a controller robust to starting states within a specified operating range.

Problem C: Bounded Uncertainty.

Problem C implements an optimal impulsive closed-loop controller in an intercept scenario. Similar to Problem B, the resulting controller is robust to a specified operating range: bounded parametric uncertainty in the dynamics for this problem.

Problems A, B, and C are summarized in Table 6.

Table 6. Summary of Problems

Problem	Configuration	Control Type	Scenario	Robust
A1	Closed-Loop	Continuous	Intercept	
A2	Open-Loop	Finite	Rendezvous	
A3	Open-Loop	Impulsive	NMC Transfer	
B	Open-Loop	Impulsive	Intercept	✓
C	Closed-Loop	Impulsive	Intercept	✓

The methodology developed and employed in this research to solve the above problems is presented in the following chapter.

III. Methodology

This chapter presents the five problem formulations and the developed methodology for their solutions. The attributes common to each problem are first presented, followed by a discussion of the unique aspects of each, primarily distinguished by the assumed control model, imposed constraints, and solution approach.

3.1 Problem Formulation and Solution

3.1.1 Dynamics.

For all problems, the HCW dynamics, discussed in § 2.2.2, are used with the mean motion chosen to be at geosynchronous orbit (except Problem A3) and dimensionless distance units (DU). The closed-form solution to the HCW equations is used to propagate the state vector when possible (e.g., impulsive thrust and unforced motion), thereby reducing computation time and ensuring accuracy. All other forced motion is numerically integrated using MATLAB's *ode45* with absolute and relative tolerances adjusted to achieve the required level of performance as indicated in each problem section. The resulting solution from the optimizer is then validated by propagating with *ode45* where applicable with finer tolerances and time steps for improved precision. The nonlinear nature of neural networks makes this validation run a vital check on the performance of the neurocontroller. Smooth (i.e., appears qualitatively continuously differentiable, C^1) and optimal performance can be verified even for network inputs, i.e. time points, that may not have been evaluated during the optimization due to the coarser time vector used. Note that throughout this thesis the control is always calculated directly from the neurocontroller, even when called within *ode45*; i.e., the neural network is a continuous function approximation of the optimal control law. Therefore, no additional interpolation is required—in other words, the neurocontroller acts as its own interpolator.

3.1.2 Decision Variables.

The method used here is at its foundation a parameterization of the control. As the function being used to parameterize the control is a fully-connected feed-forward artificial neural network, the parameters, or design variables (DV, also called decision variables), are simply the weights and biases of the network. For certain problems additional values of interest (e.g., burn duration) are parameterized as well. For sake of formulation simplicity, the nonlinear vector function property of neural networks is leveraged to parameterize all values of interest using a single network. In practice, any number of networks may be implemented for any number of desired mappings—neural networks are, if nothing else, a flexible tool. This may be especially desirable when the complexity of the individual mappings is such that it is more efficient to optimize multiple networks than to find the single network required to capture that complexity. For Problems A, B, and C, however, a single network is found to be sufficient.

Within this framework, the decision variables always consist solely of the weights and biases regardless of the network inputs and outputs; only the number of DV's change dependent on the size of the network. This abstraction conveniently allows the user to modify the underlying problem to be solved independent of the network, DV bounds, initial guesses, etc. With regards to the DV's, although each layers' associated weights and biases are most easily represented in matrix and vector form respectively, optimizers frequently require a single-column DV vectors. This remains true for the optimizers used in this research. Careful attention must be taken then to update the weights and biases at each optimization iteration/generation by expanding and reshaping this collapsed DV vector into the appropriately sized and matched matrices and vectors necessary to efficiently evaluate the neural network. As discussed in § 2.4.2, additional parameters called hyperparameters exist to specify a neural network, e.g. activation functions and topology. Although the hyperparameters may be treated as additional DV's, for this research the hyperparameters are selected based on common practice in literature [27], and through judicious trial and error. The primary hyperparameters tuned for this work are the number of hidden nodes, input/output representations and scaling, and to some extent the number of hidden layers and activation functions. This was found to be a reasonable approach given the relatively short solution time required for the problems solved, facilitating fairly quick trial and error.

With regards to parameter initialization, unfortunately there is no *a priori* insight into the appropriate values to select for the weights and biases. Common practice suggests initializing the weights from a random normal distribution and setting the biases to zero, and was the method taken in this research.[53] As well, typically a regularization term is appended to the reward function to prevent these parameters from blowing up during optimization, or in certain cases over-fitting. To avoid the careful tuning required of this term and to keep the cost function as simple as possible, the parameters are heuristically bounded to ±3 within the optimizer. Unfortunately doing so requires scaling the network input and output values appropriately, else network nodes may saturate. To mitigate this potential saturation, inputs values are normalized to the largest expected input value and output values are scaled by their expected order of magnitude. In practice the problem solution was not found to be prohibitively sensitive to the chosen scaling factors as long as the input/output values are of the order one. Selecting the scaling factors may be readily accomplished by checking if a significant number of parameters are on the bounds should the optimizer not converge.

3.1.3 Objective Functions.

In this research, the objective is to minimize control in a terminally constrained fixed-time scenario, wherein the general optimal control problem to be solved is

$$
\begin{aligned}
\underset{\boldsymbol{W},\, \boldsymbol{b}}{\text{minimize}}\ J &= \int_{t_0}^{t_f} \boldsymbol{u}(t)^T \boldsymbol{u}(t)\ dt \\
\text{s.t.} \quad \dot{\boldsymbol{x}}(t) &= \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t),\ \forall\, t \in [t_0, t_f] \\
\boldsymbol{x}(t_0) &= \boldsymbol{x}_0 \\
\delta t_0, \delta t_f &= 0 \\
\boldsymbol{x} &\in \mathbb{R}^6 \\
\boldsymbol{u} &\in \mathbb{R}^3
\end{aligned}
\tag{3.1}
$$

the cost function is the total magnitude of the control used such that the state dynamics are constrained to the specified equations of motions from a fixed initial state with no variation in the initial or final time and no variation in the final state (or subset thereof). Of course there is nothing inherent to the methodology presented here that prevents its application to other optimal control problems, (e.g. minimum and free final time, path constraints, etc.). The exact cost function formulation varies slightly for Problems A through C and depends on the control type used. For impulsive,

$$
J = \sum_{i=1}^{n} \|\Delta \boldsymbol{V}_i\|_2^2 \tag{3.2}
$$

for n total burns, where $\|\cdot\|_2$ is the 2-norm, or Euclidean norm. For finite thrust,

$$J = \sum_{i=1}^{n} \int_{0}^{t_{burn,i}} \boldsymbol{u_i}(t)^T \boldsymbol{u_i}(t)\ dt \tag{3.3}$$

where $t_{burn,i}$ is the duration of ith burn and n total burns. In some cases, e.g. fixed-thrust, Eq. (3.3) is simplified to the equivalent sum of burn durations. For continuous thrust,

$$J = \int_{t_0}^{t_f} \boldsymbol{u}(t)^T \boldsymbol{u}(t)\ dt \tag{3.4}$$

where t_0 and t_f are the initial and final times of the specific scenario respectively, as in Eq. (3.1). Numerical optimization results in a discrete control history, therefore the cost function integrals must be approximated. For this work, the integral is approximated via the trapezoidal method using MATLAB's *trapz* as it offers a simple, computationally efficient, and sufficiently accurate implementation. Each $\boldsymbol{\Delta V}$ and $\boldsymbol{u}(t)$ is a 3-by-1 control vector, consisting of components in the x, y, and z directions.

The cost functions implemented here differ from RL approaches where the cost function is typically specified as a positive reward function to be maximized; this is equivalent to minimizing the negative reward, i.e.

$$\underset{\boldsymbol{x}}{\text{minimize}}\ J(\boldsymbol{x}) = \underset{\boldsymbol{x}}{\text{maximize}}\ \ - J(\boldsymbol{x}) \tag{3.5}$$

Additionally and more importantly, typical RL approaches do not impose explicit constraints, but instead append any constraints to the reward function. This research however, seeks to leverage the existing, powerful constrained optimization tools to avoid the time-intensive task of tuning such appended constraints.

3.1.4 Hardware and Software.

Problems A, B, and C are solved on a desktop computer with a 2.4 GHz Intel Xeon E5-2640 CPU and 64 GB of RAM. Mathwork's MATLAB is used throughout this work. Problem A is solved using MATLAB's gradient-based optimization function *fmincon* and Problems B and C are solved using a combination of *fmincon* and MATLAB's genetic algorithm implementation *ga*. Both optimizers support bounded and nonlinear constraints. For *fmincon*, tolerances are adjusted to achieve desired performance with all other settings set to default values. The maximum number of generations and the population size within *ga* are adjusted by trial and error to achieve requisite convergence while all other settings are left to default values.

Multiple mature and robust machine learning frameworks exist, particularly using the Python programming language, such as TensorFlow and PyTorch. In the early stages of this research TensorFlow via the high-level Keras interface was used to explore some initial controls problems, but at the time these libraries were more conducive to supervised learning than to reinforcement learning and exhibited more complexity than required for this research. Additionally, the research team's experience was primarily in MATLAB, thereby aiding code development, as well as leveraging MATLAB's powerful optimization tools. Ultimately, MATLAB was used exclusively for the results presented here.

Although MATLAB contains its own robust neural network library, initial experimentation showed significant computational inefficiencies for the given problem formulation using MATALB's built-in neural network object class. The author was unable to determine the cause of this inefficiency. Ultimately a neural network class was developed from scratch to handle the neural network functionality required for this research to include variable number of nodes, layers, inputs, and outputs, as well as activation functions, network evaluation, and parameter updates. An additional benefit of this effort was the hands-on insight gained into neural networks that would not have been obtained with a black-box approach. Nonetheless, this author acknowledges that the existing ML tools and libraries will undoubtedly be required as the complexity and size of the problems shown here increase.

3.2 Problem A

Problem A represents a building block for Problems B and C, and of the three problems, is the most similar to traditional optimal control methods, specifically control parameterization with direct single-shooting. In terms of single-shooting [22], a neural network serves as the basis function with the weights and biases comprising the coefficients, or parameters, to be optimized over. The network hyperparameters serve as additional basis function tuning knobs to achieve convergence and the desired level of performance. As with a traditional optimal control problem, one limitation is that the resulting control is only valid for the specific problem solved. The goal is to find the optimal mapping, $\boldsymbol{\mathcal{N}}(\boldsymbol{\theta})$, in Eq. (3.6),

$$\boldsymbol{\mathcal{N}}(\boldsymbol{\theta}) : \; \boldsymbol{x} \rightarrow \boldsymbol{u}^* \tag{3.6}$$

from some input vector $\boldsymbol{x}$ to an optimal output vector $\boldsymbol{u}^*$, where $\boldsymbol{\theta}$ is the parameter vector comprising the neural network weights ($\boldsymbol{W}$) and biases ($\boldsymbol{b}$). If $\boldsymbol{x}$ represents the initial state, $\boldsymbol{x}_0$, the mapping is only valid, that is optimal, for the initial state chosen. To find the optimal control from a different initial state requires re-solving the entire problem again. An alternative approach to overcome this limitation is presented in Problems B and C.

Nonetheless, this method is taken as an initial feasibility study for the development of optimal neurocontrollers in the proximal spacecraft maneuver domain.

Multiple control models and configurations are explored and the relative complexity of the problems ascertained. The three sub-problems within Problem A illustrate the application of neural networks to a variety of control models, configurations, and scenarios, and provides insight into the requisite networks to be used as an initial point of reference in Problems B and C.

3.2.1 Problem A1: Intercept.

For this problem, a continuous control is implemented in a closed-loop configuration and applied to an intercept scenario. The spacecraft must match its final position with a designated target position, while its final velocity is allowed to be free, i.e.,

$$\begin{aligned} &\boldsymbol{r}(t_f) = \boldsymbol{r}_{des}(t_f) \\ &\delta \boldsymbol{v}(t_f) \text{ is free} \end{aligned} \tag{3.7}$$

Figure 11 illustrates the network employed for this problem, with the 6-by-1 current state vector taken as input and a 3-by-1 acceleration vector as output. Algorithm 1 outlines the approach taken. Optimization is accomplished with *fmincon* and *ode45* is used to integrate the forced motion dynamics.

Algorithm 1 Closed-Loop Control Optimization

```
def $\mathbf{x}(t_0),\ \mathbf{x}_{des}(t_f),\ \mathbf{t} = [t_0, t_f]$
init guess $\boldsymbol{\theta}$
procedure OPTIMIZE LOOP
    update $\boldsymbol{\theta}$
    $\mathbf{X},\ \mathbf{U} \leftarrow \mathbf{propagate}\big(\mathbf{x}(t_0), \mathbf{t}, \mathbf{u}(\boldsymbol{\theta}, \mathbf{x}(t))\big)$
    $J \leftarrow \mathbf{cost}(\mathbf{U})$
    $\mathbf{c} \leftarrow \mathbf{constraints}\big(\mathbf{x}(t_f), \mathbf{x}_{des}(t_f)\big)$
return $J,\ \mathbf{c},\ \boldsymbol{\theta}$
```

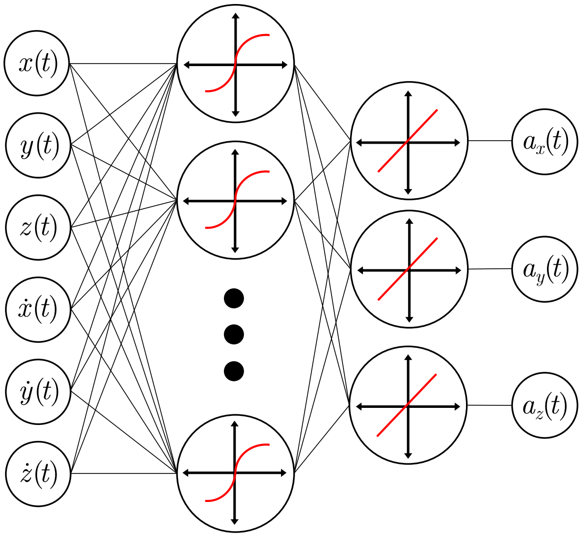


Figure 11. Problem A1 Network

3.2.2 Problem A2: Rendezvous.

For this problem, a finite thrust controller is implemented in an open-loop configuration for a rendezvous scenario. The spacecraft must match final position and velocity with the desired values, i.e.,

$$\begin{aligned} \boldsymbol{r}(t_f) &= \boldsymbol{r}_{des}(t_f) \\ \boldsymbol{v}(t_f) &= \boldsymbol{v}_{des}(t_f) \end{aligned} \tag{3.8}$$

The controller is configured for a burn-coast-burn trajectory with each burn specified as fractions of half the total trajectory time. Network inputs are a 6-by-1 initial state vector with 8 total outputs: two 3-by-1 control direction vectors and two scalar burn durations. The direction of the burns are constant over their duration with a predefined fixed thrust magnitude. Unlike in Problem A1, the open-loop configuration in Figure 12 allows the control to be evaluated prior to propagation, as indicated in line 5 of Algorithm 2.

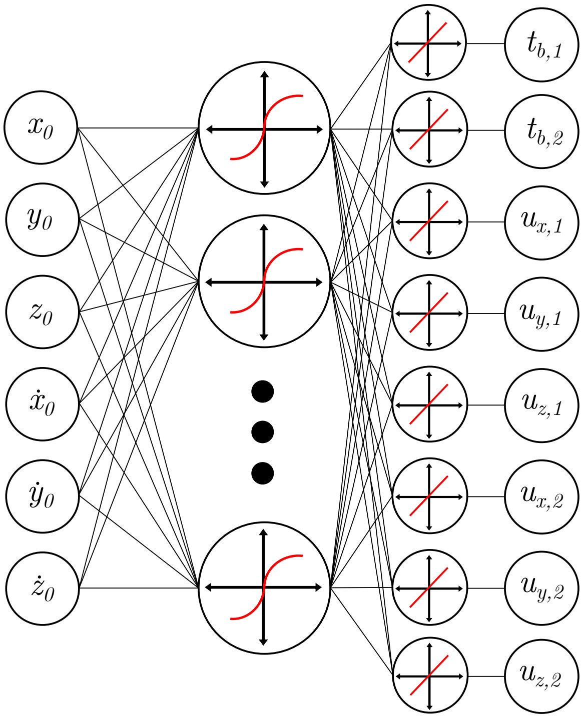


Figure 12. Problem A2 Network

Algorithm 2 Open-Loop Control Optimization

```
def $\mathbf{x}(t_0),\ \mathbf{x}_{des}(t_f),\ \mathbf{t} = [t_0, t_f]$
init guess $\boldsymbol{\theta}$                                      ▷ weights & biases
procedure OPTIMIZE LOOP
    update $\boldsymbol{\theta}$
    $\mathbf{U} = \mathbf{u}\big(\boldsymbol{\theta}, \mathbf{t}, \mathbf{x}(t_0)\big)$          ▷ entire control trajectory
    $\mathbf{X} \leftarrow \mathbf{propagate}(\mathbf{x}(t_0), \mathbf{t}, \mathbf{U})$
    $J \leftarrow \mathbf{cost}(\mathbf{U})$
    $\mathbf{c} \leftarrow \mathbf{constraints}(\mathbf{x}(t_f), \mathbf{x}_{des}(t_f))$
return $J,\ \mathbf{c},\ \boldsymbol{\theta}$
```

3.2.3 Problem A3: NMC Transfer.

For Problem A3, an impulsive thrust controller is implemented in an open-loop configuration for an NMC transfer scenario. Whereas the previous two problems passed the state vector to the network, here the network inputs are the 6 initial ROE's. Although equivalent results were achieved using the initial state vector, this representation further illustrates the flexibility of neurocontrollers. Problem B demonstrates a case whereby equivalent performance is not as easily obtained with both representations. For the network outputs, two 3-by-1 ΔV vectors for the initial and final burns make up the 6 output nodes as depicted in Figure 13. In the case of an NMC transfer, the terminal constraints are

$$\begin{aligned} a_e(t_f) &= a_{e,des}(t_f) \\ x_d(t_f) &= x_{d,des}(t_f) \\ y_d(t_f) &= y_{d,des}(t_f) \\ z_{max}(t_f) &= z_{max,des}(t_f) \\ \psi(t_f) &= \psi_{des}(t_f) \end{aligned} \tag{3.9}$$

thereby ensuring the desired relative orbital geometry is achieved. The final phase angle along this orbit is allowed to be free. The method used for solving this problem is similar to the method outlined in Problem A2.

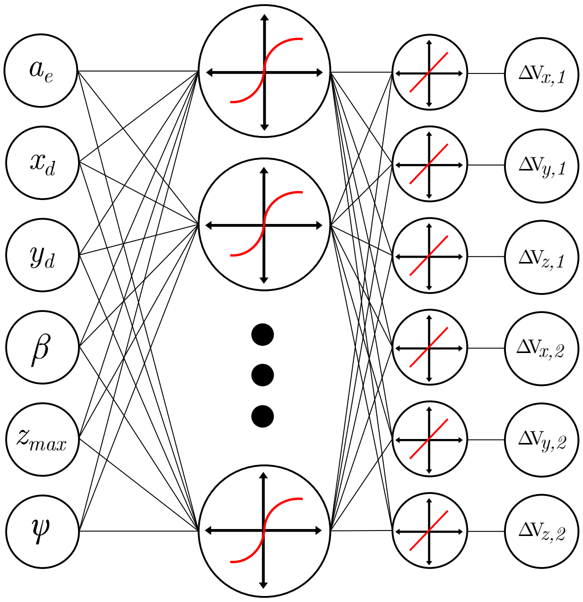


Figure 13. Problem A3 Network

3.3 Problem B: Bounded Initial Conditions

Problems A1-A3 are similar to traditional optimal control methods in that they produce solutions for a single scenario—change any of the initial conditions, position for example, and the control is no longer optimal or valid. A generalized optimal controller is sought however, akin to the mapping in Eq. (3.10) than Eq. (3.6). Problems B and C attempt to find a generalized mapping.

$$\mathfrak{N}(\boldsymbol{\theta}) : \; \forall \, \boldsymbol{x}(t_0), \boldsymbol{x}_{des}(t_f) \in [\boldsymbol{x}_{lb}, \boldsymbol{x}_{ub}] \rightarrow \boldsymbol{u}^*(t) \; \forall \, t \in [t_0, t_f] \tag{3.10}$$

This generalization property represents a real strength of neural networks, whereas in traditional optimization the control variables are solved for directly, neural networks provide a level of abstraction to produce more generalized controllers. This is also why Problems A1-A3 have more network inputs than seemingly necessary, and in the continuous control case, could have been optimized using time as the sole input, as is commonly the case in direct single-shooting methods.[22] However, by incorporating the additional inputs from the start, the transition to the generalized controller construct did not require any significant modifications to the existing network structures, and to a certain extent, provided some initial insight into the requisite network hyperparameters.

For Problem B, an impulsive controller in an open-loop configuration is implemented and applied to an intercept scenario. As in Problem A2, the network inputs are the initial state vector and similar to Problem A3, the network outputs are a 3-by-1 ΔV control vector. Unlike the intercept scenario of Problem A1 however, an operating range is defined as any starting position along a specified NMC, that is for all $\beta_0 \in [0, 2\pi]$, as depicted by the dashed line in Figure 15. In the ML and RL community this is referred to as the environment.[27] The goal here then is to produce

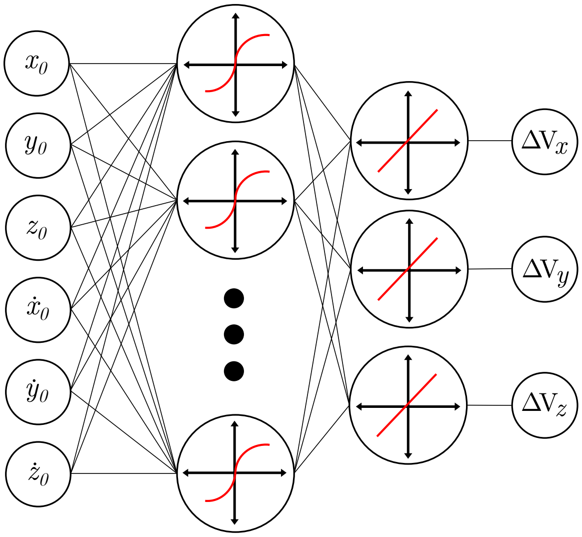


Figure 14. Problem B Network

a single neurocontroller capable of generating the optimal impulsive burn regardless of starting position within the defined operating range.

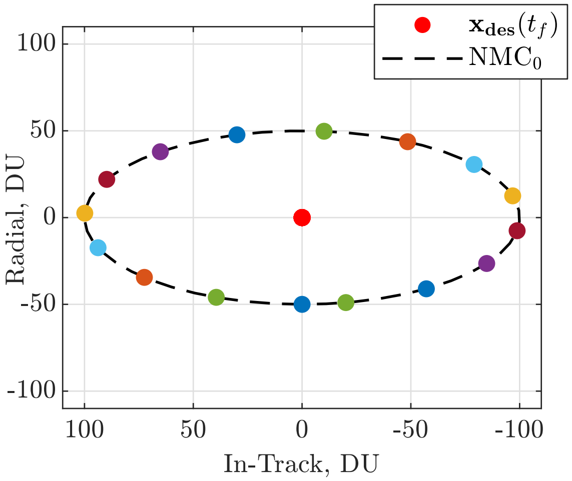


Figure 15. Problem B Operating Range and Training Set

Two approaches to solve this problem are explored, each dependent on sampling a subset, or training set (depicted notionally by the outer dots in Figure 15), of the full operating range: (a) at every iteration of the optimization, simulate the controller on all samples, or (b) at each iteration, simulate a smaller sample subset, or batch, of the total training set. Approach (a) is amenable to gradient-based optimization, but does not scale well as the size of the training set or complexity of the problem increases. Problems B and C remained within the scope of this approach however, and combinations of both approaches were explored. In current ML/RL research however, approach (b), and variations thereof, is most commonly taken. Fundamentally, at each iteration (also called episode) of the optimizer, a batch of samples from the training distribution is randomly selected, then simulated using the candidate controller, and finally the cost function taken to be some statistic of the batch (e.g., mean, max, etc.). Over the course of the entire optimization, or training process, a representative portion of the entire distribution will have been sampled.

Algorithm 3 Evolutionary Network Optimization with Batching

```
def environment, $\mathcal{E}$                                   ▷ objective, constraints, dynamics, etc.
def agent, $\mathcal{G}$                                   ▷ network topology, activation functions, etc.
procedure OPTIMIZE
    init population of $\mathcal{G}$
    while stop criteria not met do
        for each candidate $\mathcal{G}$ do                                   ▷ in parallel
            for each sample do
                randomly sample $\mathcal{S}(t_0)$ and $\mathcal{E}$ distributions               ▷ batch
                while scenario not done do
                    $\mathcal{A} \leftarrow \mathcal{G}(\mathcal{S})$                              ▷ determine action
                    $\mathcal{S}' \leftarrow \mathcal{E}(\mathcal{S}, \mathcal{A})$                    ▷ propagate, get new state
                    $t = t + \Delta t$
        return constraints and mean cost
        evolve population
    return best $\mathcal{G}$
validate
```

The stochastic nature of this approach can pose challenges to some gradient-based optimizers. Typically RL problems are formulated to avoid explicit constraints, but for this paper terminal constraints are included. Heuristic methods are chosen, specifically a genetic algorithm, to accommodate both the cost and nonlinear constraint functions directly. Algorithm 3 and Figure 16 outline the general framework. Random permutations are used for batch selection to avoid duplicate samples within the

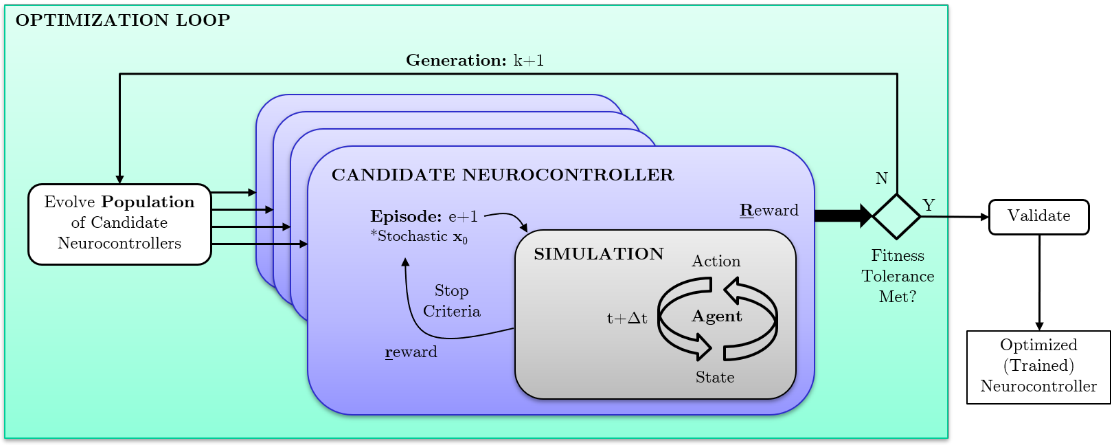

OPTIMIZATION LOOP
Generation: k+1
Evolve Population of Candidate Neurocontrollers
CANDIDATE NEUROCONTROLLER
Episode: e+1
*Stochastic $\mathbf{x}_0$
Reward
SIMULATION
Action
Agent
t+Δt
State
Stop Criteria
reward
N
Y
Fitness Tolerance Met?
Validate
Optimized (Trained) Neurocontroller

Figure 16. Evolutionary Network Optimization Framework

same iteration/generation. Selecting the right number of samples depends on the size and variance of the training set, as well as the complexity of the problem itself, and is an active area of current research.[27] For this work, these values are determined by trial and error.

With the addition of terminal constraints, it was found that tightening the tolerances over multiple iterations avoids premature divergence of the optimizer. In other words, the optimizer is first allowed to find a feasible solution in the neighborhood of the optimal solution, and then gradually increase the terminal accuracy and optimality of that solution. This is accomplished with a variation of the logistics function according to

$$\epsilon_i = \epsilon_d(1 + 10e^{-\gamma i}) \tag{3.11}$$

where ϵ_i is the tolerance at iteration i, ϵ_d is the desired final tolerance, and γ is a user-defined growth rate. The logistics function was chosen through trial and error as a common and convient method for tightening the tolerance. After convergence, an additional optimization pass is made using all samples and finally validated on a larger sampling within the training set bounds to validate generalization performance and to verify smoothness such that no undesirable nonlinearities exist.

An additional benefit of neural networks may be observed in this open-loop control implementation by leveraging the vector, in fact matrix, property of neural networks.[27] The optimal control for a range of β_0 values may be calculated directly and efficiently in one evaluation of the network. Within Problem B, this may be accomplished by passing to the network a 6-by-N matrix of initial states corresponding to the desired 1-by-N β_0 vector. The network output is then the 3-by-N ΔV matrix corresponding to each initial state. Alternatively, if an open-loop continuous controller is implemented with time as input, the entire control history may be similarly calculated by passing the desired time vector. The matrix aspect of neural networks is one property contributing to their popularity; as the size and complexity of the problem increases this benefit becomes more pronounced, with hardware dedicated to matrix calculations further improving efficiency. Unfortunately this can only be accomplished for open-loop configurations, as closed-loop controllers exhibit a recursive dependency on states not known *a priori*, i.e., network output $\boldsymbol{u}(t_i)$ depends on network input $\boldsymbol{x}(t_i)$ that itself must first be propagated using $\boldsymbol{u}(t_{i-1})$, where t_i is the ith time step.

3.4 Problem C: Bounded Uncertainty

Similar to the previous problem, a generalized controller is sought, and in this case one that performs optimally despite some uncertainty in the state dynamics. An uncertain relative perturbation term, α, is introduced in the in-track direction, see Eq. (3.12), and assumed to be constant with known bounds, $[0.1, 1]$. Once again, an impulsive controller is implemented for an intercept scenario but due to the uncertainty, the controller uses a closed-loop configuration.

$$\ddot{y} = -2n\dot{x} - \alpha \cdot 10^{-8} \tag{3.12}$$

Equation 3.13 shows the desired mapping.

$$\mathfrak{N}(\boldsymbol{\theta}) : \ \boldsymbol{x}(t_0), \boldsymbol{x}_{des}(t_f) \rightarrow \boldsymbol{u}^*(t) \ \forall \ \alpha \in [0.1, 1] \tag{3.13}$$

The bounds on α were selected such that a neurocontroller trained on an environment with $\alpha = 0$ results in a terminal position error of approximately 1 DU and 10 DU for actual α values of 0.1 and 1 respectively. Therefore, the chosen values for α are significant enough to impact the dynamics and should not be neglected by the controller. The network took as an input the 6-by-1 current state vector and parameter estimate $\hat{\alpha}$, and output a 3-by-1 ΔV vector as well as an updated $\hat{\alpha}$, depicted in Figure 17. Following the batched approach (Algorithm 3) in Problem B, a training set of 50 linearly spaced α values is generated within the specified bounds. The number of samples at each iteration was 5. The initial parameter guess, $\hat{\alpha}_0$, was set in the middle of the distribution at 0.5. The tolerance growth factor γ in Eq. (3.11) is set to 1.

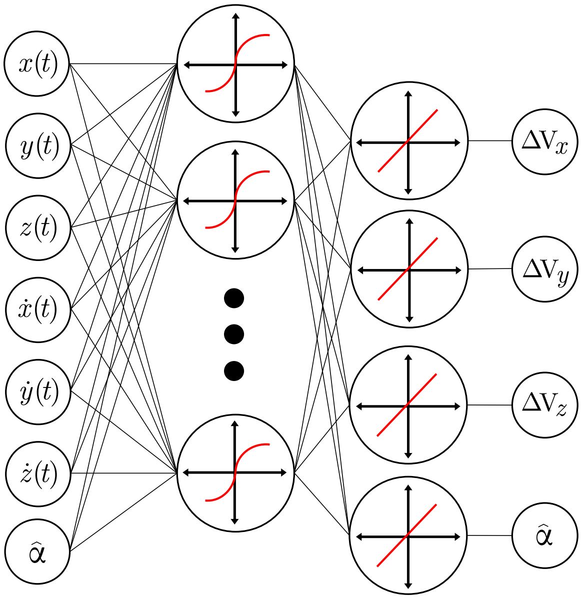


Figure 17. Problem C Network

The trajectory was divided into 3 fixed-length coast phases. At the end of coast 1, allowing the network time to estimate the uncertainty, the network is passed the current state vector $\boldsymbol{x}(t_{coast,1})$ and $\hat{\alpha}_0$, and outputs a new $\hat{\alpha}$. The output ΔV is ignored on this pass. The network is then passed the updated $\hat{\alpha}$ and same $\boldsymbol{x}(t_{coast,1})$, and the output ΔV is executed and $\hat{\alpha}$ updated. The second coast phase is propagated using the closed-form solution, then $\hat{\alpha}$ and $\boldsymbol{x}(t_{coast,2})$ are passed to the network, the output ΔV is executed and the final coast phase propagated. Note that the value of $\hat{\alpha}$ does not necessarily correspond to the actual α value as this is not enforced during the optimization, but is nonetheless correlated. A variation of this setup could enforce $\hat{\alpha} = \alpha$ as a constraint within the optimization, or split the parameter estimation out to a second network, thereby creating a dual-network controller similar to the structure of traditional estimator-regulator controllers.[54]

The following chapter details the implementation of Problems A, B, and C for specific sets of values, and for each, presents the results alongside a benchmark optimal solution.

IV. Implementation and Analysis

This chapter presents the specific implementation of each of the five problems within this research for a given set of parameter values. The results are then presented and discussed with a comparison to benchmark solutions.

4.1 Problem A

4.1.1 Problem A1: Intercept.

For all problems, the initial spacecraft state is specified using ROE's. For the first problem, the following initial values are used

Table 7. Problem A1 Initial ROE's

a_e	x_d	y_d	β	z_{max}	ψ
100	0	0	45°	50	270°

resulting in an initial state vector

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{r} \\ \boldsymbol{v} \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \approx \begin{bmatrix} -35 \\ 71 \\ -50 \\ 0.003 \\ 0.005 \\ 0 \end{bmatrix} \text{DU, DU/s}$$

with a fixed final time of $t_f = 1/4\,\text{orbit}$. The origin is specified at an altitude of 35 786 km.

To prevent network saturation, the network inputs are scaled to approximately order one: position values by $1 \cdot 10^{-2}$ and velocity values by 100. The network outputs, in this case the control values, are scaled by $1 \cdot 10^{-7}$ such that the network output values are on the order one. The terminal position constraint is implemented as a nonlinear inequality constraint,

$$\|\boldsymbol{r}(t_f) - \boldsymbol{r}_{des}(t_f)\|_2^2 \leq \epsilon_r^2 \tag{4.1}$$

where $\boldsymbol{r}_{des} = 0$ for this example, i.e. the origin, and ϵ_r is the user-specified tolerance, for this example $1 \cdot 10^{-3}$ DU. The cost function uses a trapezoidal approximation of the control integral and is scaled appropriately.

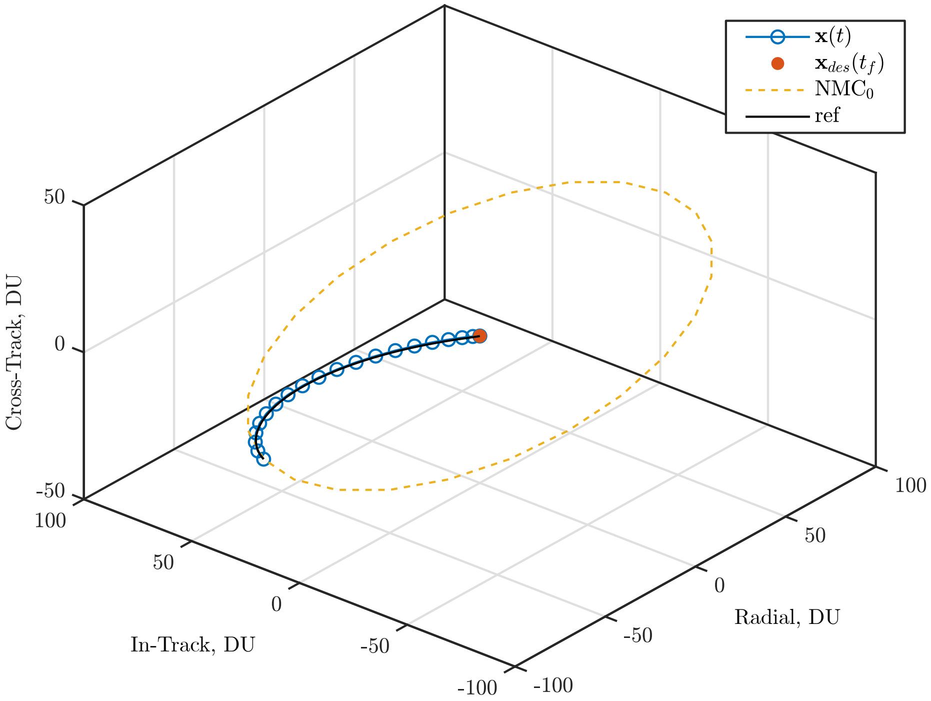


Figure 18. Problem A1 Trajectory

The desired level of performance is obtained with only 4 hidden nodes, resulting in 36 weights and 7 biases for a total of 43 parameters. The final parameter values are summarized in Figure 19, with the size of each node corresponding to its relative bias, the opacity of the connections corresponding to the relative weights, and positive and negative values represented by green and red respectively. Within this research these plots are primarily used to determine the appropriate network output scale factor. In this limited case however, some additional insights may be gleaned by comparing Figure 19 to the control history in Figure 20. The relative magnitude and sign of the output nodes, i.e. the bias values, roughly correspond to the magnitude and sign of the corresponding control components. As well, the connections leading to the output node corresponding to the z control component are relatively transparent indicating those weights are near zero. This matches the control plot that shows the z component is nearly always zero. From these observations it is hypothesized that the output layer biases are the primary driving factors of the control while weights and biases to the left in this diagram serve to fine tune the control. Additionally, a significant pathway is observed from the $\dot{y}$ input node to the a_x output node, suggesting there may be a strong relation between these two components in this problem. The author is hesitant to draw any definitive conclusions however from observations of these network plots. More research is required to extract any consistently meaningful patterns beyond this limited context.

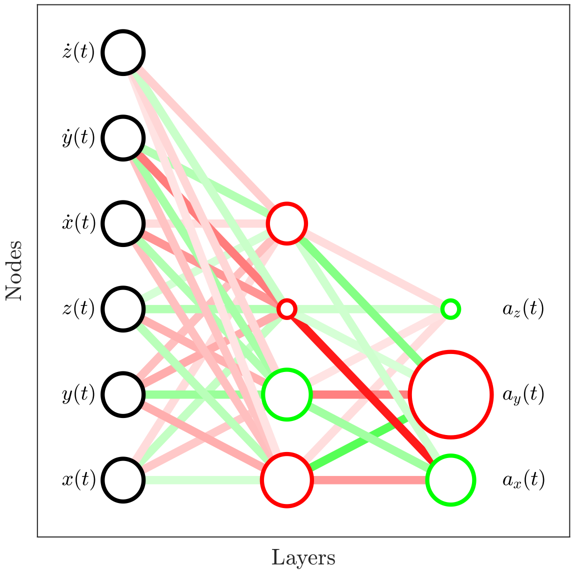


Figure 19. Problem A1 Network Weights/Biases

An optimal benchmark solution was found using GPOPS-II (a commonly used and mature MATLAB-based pseudospectral optimization tool). The resulting trajectories are depicted in Figures 18-21, with the GPOPS-II solution indicated by the dashed lines. The neurocontroller results compare favorably to the optimal reference. As integration within the optimization is carried out using a fairly coarse time step for computational performance, it is necessary to verify the results of the neurocontroller at a finer time step due the potentially highly nonlinear nature of the network function. From this verification step, the control is observed to be smooth and without any unexpected nonlinearities.

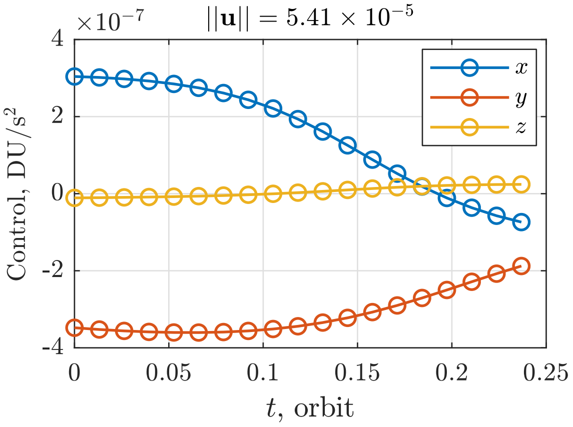


Figure 20. Problem A1 Control History

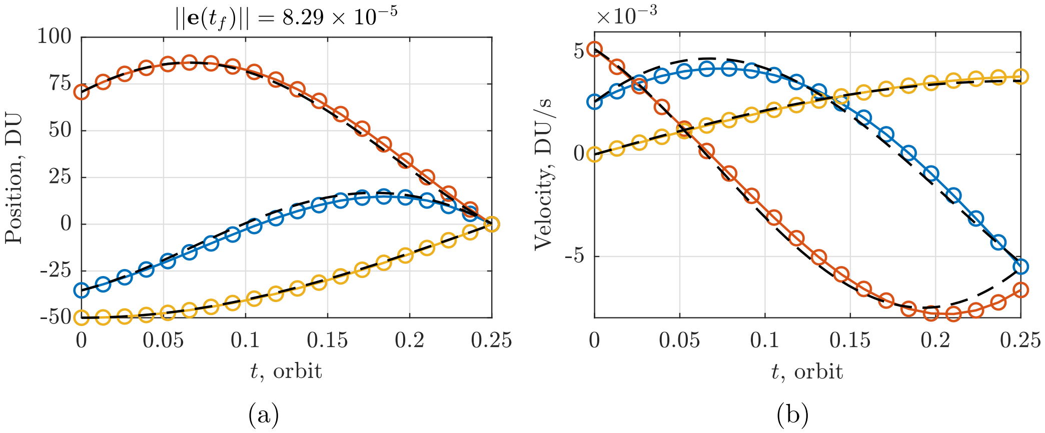


Figure 21. Problem A1 State History

4.1.2 Problem A2: Rendezvous.

For Problem A2 the following initial values are used

Table 8. Problem A2 Initial ROE's

a_e	x_d	y_d	β	z_{max}	ψ
100	0	0	0	0	0

resulting in an initial state vector

$$\boldsymbol{x} \approx \begin{bmatrix} -50 & 0 & 0 & 0 & 0.007 & 0 \end{bmatrix}^T \text{DU, DU/s} \tag{4.2}$$

with a fixed final time of $t_f = 1/4\,\text{orbit}$ and the origin specified at an altitude of 35 786 km.

The network inputs are scaled by the same values as in Problem A1, $1 \cdot 10^{-2}$ and 100 for position and velocity respectively. The thrust magnitude is fixed at $1 \cdot 10^{-6}\,\text{DU/s}^2$, therefore the network is only solving for the thrust direction and duration of the two finite burns. No scaling is applied to the network outputs. The burn durations are determined by mapping the corresponding network output, $\mathcal{N}_{t_{b,i}}$, to the range $[0, 1]$ using the sigmoid function and multiplying by the maximum burn duration, $t_{b,max}$, i.e.,

$$t_{b,i} = t_{b,max} \cdot \text{sigmoid}(\mathcal{N}_{t_{b,i}}), \quad t_{b,max} = \frac{t_f}{2} \tag{4.3}$$

This can be readily generalized to n burns by making $t_{b,max} = \frac{t_f}{n}$. In this manner, the entire trajectory time may be used for thrust, but each burn is nonetheless limited by these maximum uniform segments. The coast time is simply the remaining time, $t_f - \sum t_b$. Similarly, the thrust directions are determined by taking the unit vector of the 3 corresponding network outputs, $\mathcal{N}_{\boldsymbol{a}_i}$, and scaling by the thrust magnitude,

T, i.e.,

$$a_i = T\frac{\boldsymbol{\mathcal{N}}_{\boldsymbol{a}_i}}{\|\boldsymbol{\mathcal{N}}_{\boldsymbol{a}_i}\|} \tag{4.4}$$

As in Problem A1, the terminal constraints are implemented as nonlinear inequality constraints, such that

$$\begin{aligned} \|\boldsymbol{r}(t_f) - \boldsymbol{r}_{des}(t_f)\|_2^2 &\leq \epsilon_r^2 \\ \|\boldsymbol{v}(t_f) - \boldsymbol{v}_{des}(t_f)\|_2^2 &\leq \epsilon_v^2 \end{aligned} \tag{4.5}$$

where once again the desired final state in this example is the origin with zero velocity. The tolerance ϵ_r remains at $1 \cdot 10^{-3}$ DU while ϵ_v is introduced with a value of $1 \cdot 10^{-6}$ DU/s. The velocity constraints are scaled by $1 \cdot 10^{-3}$ to ensure both constraints are of approximately equivalent magnitudes. Unlike Problem A1, the fixed thrust magnitude allows the cost function to be easily calculated from the burn durations, such that

$$J = \sum_{i=1}^{n} t_{b,i} \tag{4.6}$$

The open-loop configuration in this problem obviates the need to perform any finer time step validation, outside of any desired numerical precision, on the neurocontroller results as was accomplished in Problem A1.

As in Problem A1, acceptable performance is obtained with only four hidden nodes, albeit the additional outputs increase the total number of parameters to 68, summarized in Figure 22. The neurocontroller is compared to an optimal benchmark solution found using a direct static optimization formulation with *fmincon* where the design variables are explicitly burn durations and direction vectors. Both results are illustrated in Figures 23-24. The two state histories match favorably with some discrepancy towards the end due to the second burn differences. While the direction and duration of the first burn is nearly identical between the two methods, for the

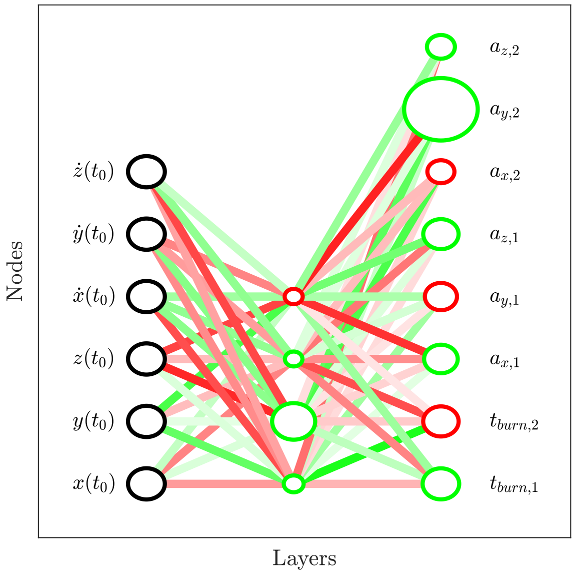
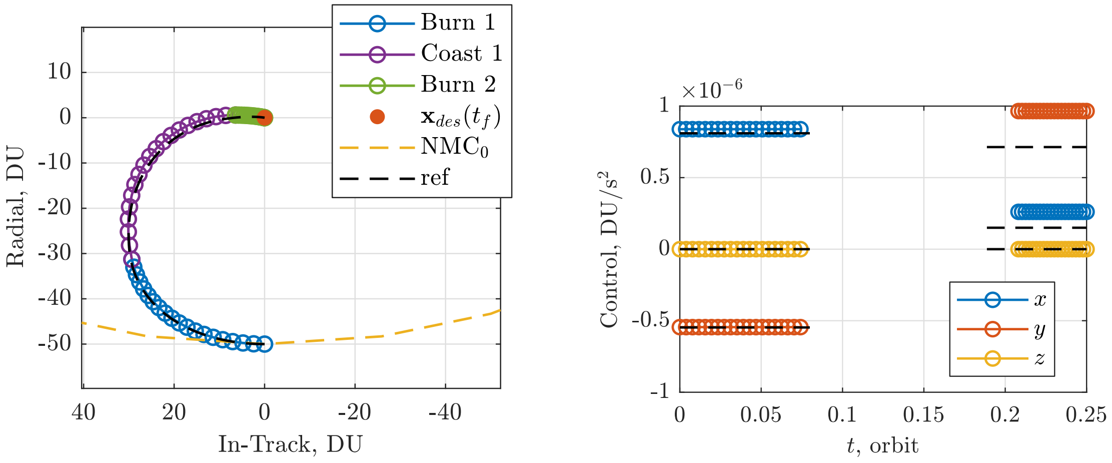


Figure 22. Problem A2 Network Weights/Biases



(a) Trajectory

(b) Control

Figure 23. Problem A2 Trajectory and Control

second burn the neurocontroller burned for a shorter duration at a higher magnitude compared to the benchmark. The resulting total control are approximately equal, the discrepancy in the latter burn suggesting a non-unique solution exists.

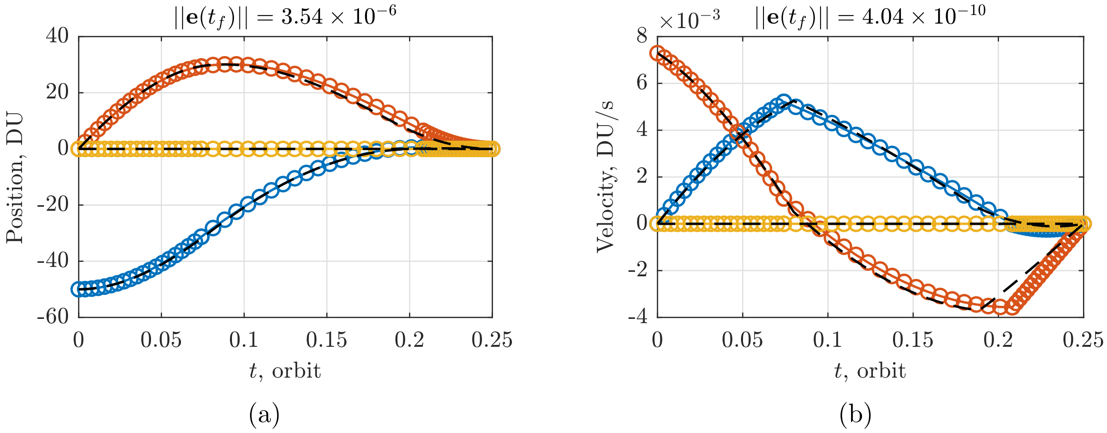


Figure 24. Problem A2 State History

4.1.3 Problem A3: NMC Transfer.

For Problem A3 the following initial and desired final values are used:

Table 9. Problem A3 ROE's

	a_e	x_d	y_d	β	z_{max}	ψ
Initial	100	0	0	90°	0	0
Desired	200	0	0	free	0	0

with a fixed final time of $t_f = 1/2$ orbit. The origin is specified at an altitude of 622 km. These values are chosen to match Lovell's analytical development in [17].

Unlike Problems A1 and A2, the network inputs for this problem are the ROE's, therefore all position values are scaled by $2 \cdot 10^{-2}$, both angles by $\frac{1}{2\pi}$, and network outputs by $1 \cdot 10^{-3}$. The terminal position constraint is separated into two nonlinear inequality constraints, one for the position terms

$$\|\boldsymbol{r}(t_f) - \boldsymbol{r}_{des}(t_f)\|_2^2 \leq \epsilon_r^2 \tag{4.7}$$

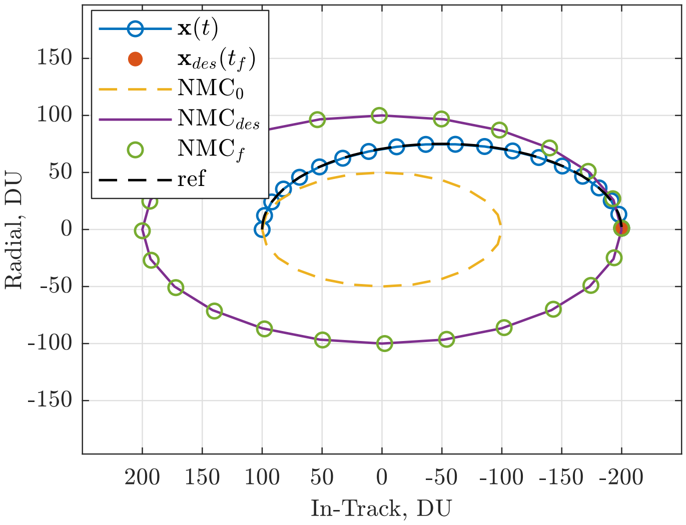


Figure 25. Problem A3 Trajectory

where in this context $\boldsymbol{r}$ comprises the ROE's a_e, x_d, y_d, and z_{max}, and one constraint for the angle

$$\|\psi(t_f) - W(\psi_{des}(t_f))\|_2^2 \leq \epsilon_\psi^2 \tag{4.8}$$

where $W(\cdot)$ simply wraps its argument to the range $[0, 2\pi]$. Both tolerances are set to $1 \cdot 10^{-3}$. The state vector at the final time, $\boldsymbol{x}(t_f)$, is transformed to ROE's using Eq. (2.5). In this case, the final β is allowed to be free, i.e., the spacecraft can enter the desired final NMC anywhere along the ellipse. The cost function is the total $\Delta\boldsymbol{V}$,

$$\sum_{i=1}^{n} \|\Delta\boldsymbol{V}_i\|_2^2 \tag{4.9}$$

As the minimum control for this scenario requires only radial burns, it is found that with 4 hidden nodes as in the previous two problems, non-zero in-track and cross-track burn components are output by the network, resulting in a slightly higher overall ΔV than optimal. By increasing the number of hidden nodes to 12, the in-track and cross-track components are driven to zero, producing the optimal total ΔV found in [17]. Note that although the final β value is allowed to be free in this formulation, the final β specified in [17] represents the minimum control solution, therefore, the same trajectory is arrived at regardless. Overall, this network comprises 160 total parameters, summarized in Figure 26. Both the neurocontroller and results found in [17] are depicted in Figures 25 and 27. Note the $\dot{x}$ component discrepancy observed in Fig. 27b is simply an artifact of the difference in time steps between the two trajectories.

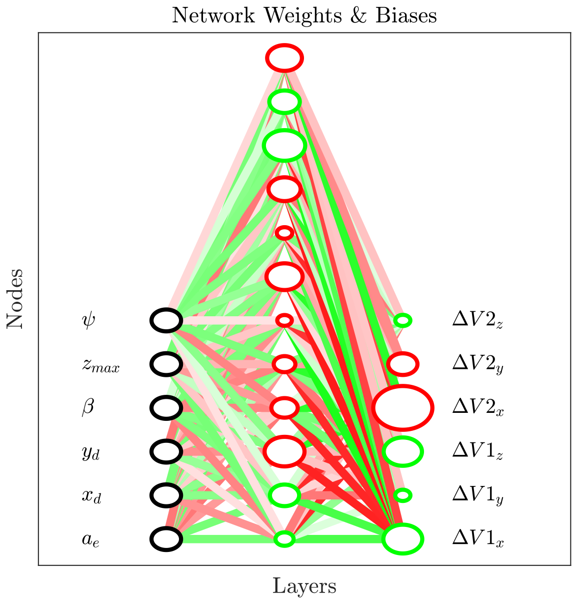
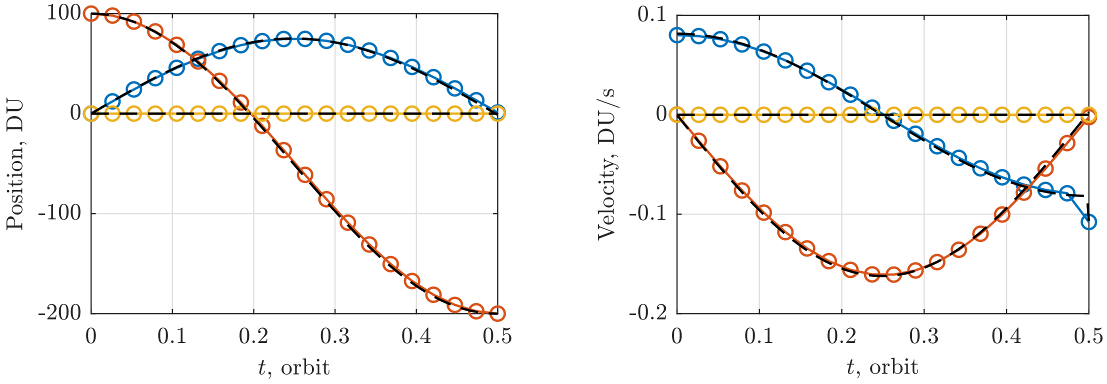


Figure 26. Problem A3 Network Weights/Biases



(a)

(b)

Figure 27. Problem A3 State History

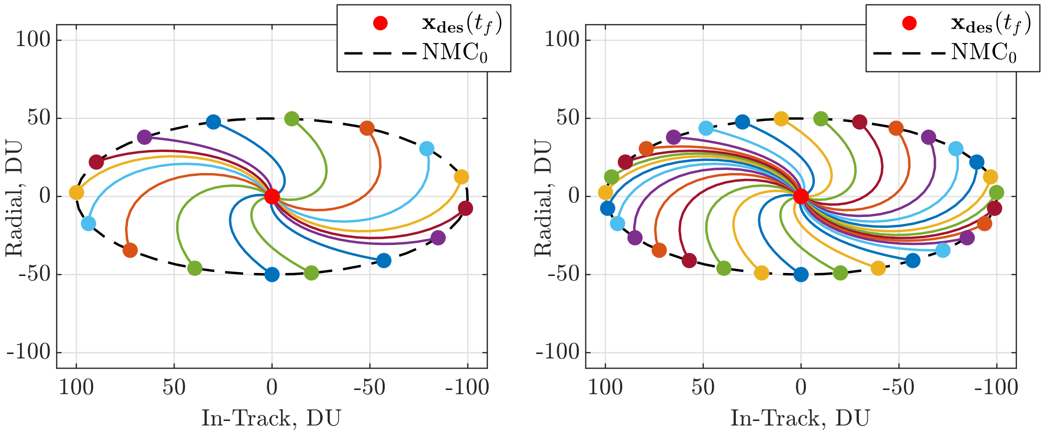
4.2 Problem B: Bounded Initial Conditions

For Problem B, the following initial values are used:

Table 10. Problem B Initial ROE's

a_e	x_d	y_d	β	z_{max}	ψ
100	0	0	$[0, 2\pi]$	0	0

with a fixed final time of $t_f = 1/4$ orbit. The origin is specified at an altitude of 35 786 km.



(a) Trajectories from training set initial states

(b) Trajectories from initial states outside training set

Figure 28. Validation of generalized performance beyond 16 β_0 **values**

Since β_0 is the only varying input in this problem, initially a network was developed with β_0 as the only input. With only one input, the total number of network parameters is reduced thereby improving computational efficiency. However, it was found that an equivalent network (i.e., same number of hidden nodes) with the full 6-by-1 initial state vector as the input performed better with regards to optimality, accuracy, and convergence. Thus the network inputs are scaled as in Problems A1 and A2, $1 \cdot 10^{-2}$ and 100 for position and velocity respectively. The network output is scaled by $1 \cdot 10^{-3}$.

Initially the same terminal position constraint as in Problem A1, Eq. (4.1) was used. This constraint, however, results in premature divergence in the *ga*. When run for a single β_0 value, convergence is attained. This suggests the stochastic nature of the batching presents a challenge to the optimizer. To overcome this challenge, the method outlined in Eq. (3.11) is implemented whereby the constraint is gradually tightened over multiple generations of the *ga*. The constraint becomes

$$\|\boldsymbol{r}(t_f) - \boldsymbol{r}_{des}(t_f)\|_2^2 \leq \left(\frac{\epsilon_r}{1 + 10e^{-\gamma i}}\right)^2 \tag{4.10}$$

For this case, a γ value of 0.1 is found to be acceptable with $\epsilon_r = 1 \cdot 10^{-3}$ DU. This results in a constraint vector equal in length to the number of samples, s, in the batch. The cost function is the mean square $\Delta\boldsymbol{V}$,

$$J = \frac{1}{s}\sum_{i=1}^{s} \|\Delta\boldsymbol{V}_i\|_2^2 \tag{4.11}$$

The training set comprises 16 linearly spaced values of β_0 in the defined range $[0, 2\pi]$. The batch size is gradually incremented from 4 to 8 and finally the full 16. These values are tuned by hand to achieve convergence at minimum computation time. The final pass is made with the full training set using *fmincon* at the specified tolerance to ensure a local minimum is reached from the seeded *ga* solution. In this case, only four hidden nodes (43 total parameters) are required to achieve the performance illustrated. Figure 28a depicts the trajectories for 16 initial β_0 values in the training set, matching the optimal trajectories. Figure 28b depicts 16 additional trajectories for initial β_0 values outside the training set and validates the controller's ability to generalize optimally beyond the 16 trained samples. As well, the blue-red double trajectory at perigee shows the network suitably handles $\beta_0 = 0 \equiv 2\pi$. Figure 30 shows the NC results for a further 64 initial β_0 values compared to an optimal benchmark solution.

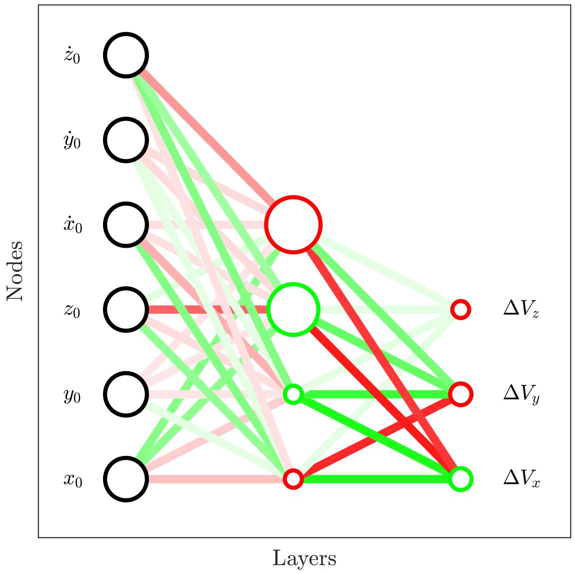
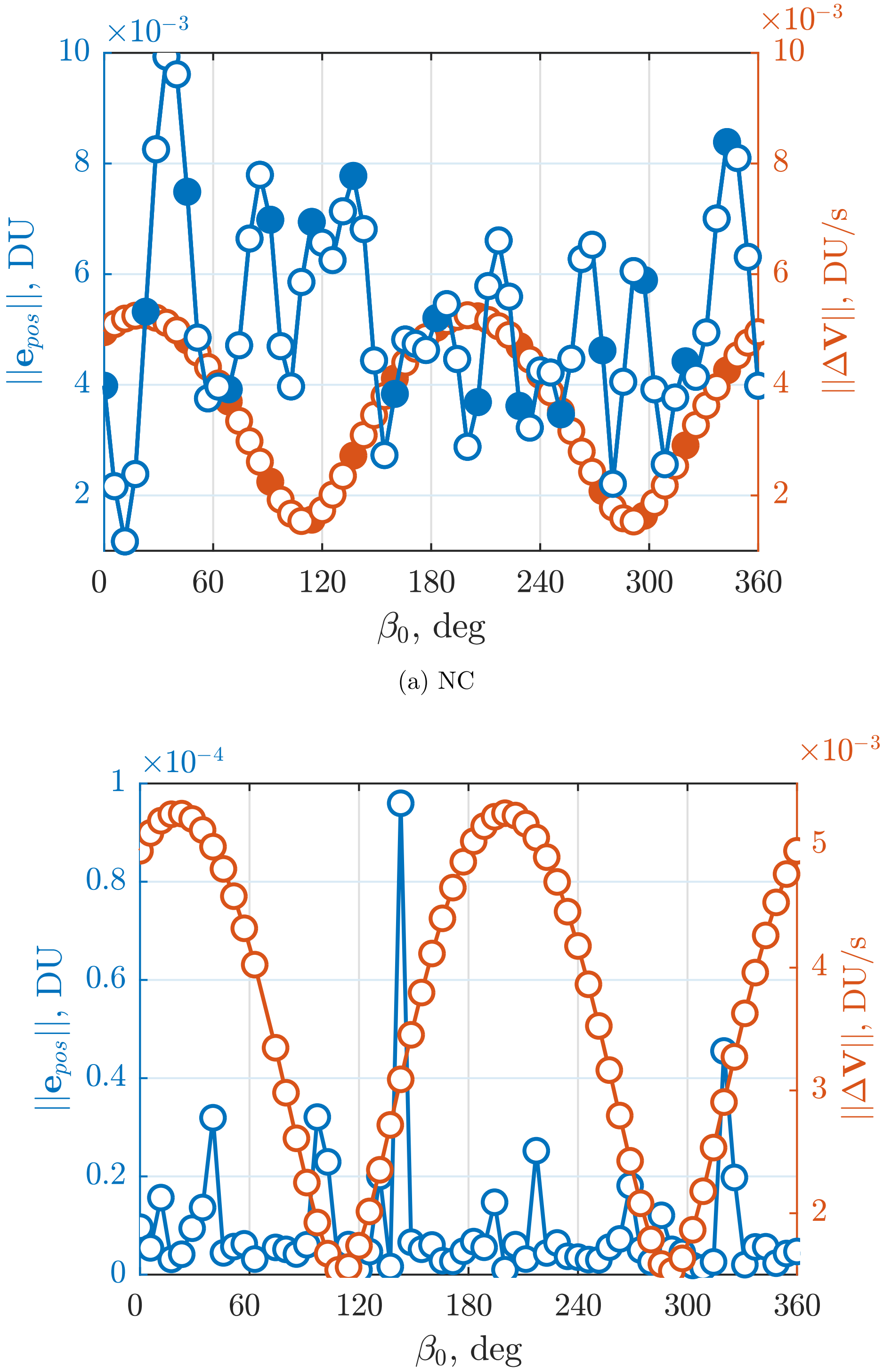


Figure 29. Problem B Network Weights/Biases

The benchmark was found by solving 64 individual static optimization problems with *fmincon*, corresponding to the 64 β_0 values. The filled circles represent the 16 point training set. The NC performance favorably matches the optimal control costs albeit with a slightly higher terminal position error, the latter of which may be improved with a deeper network.

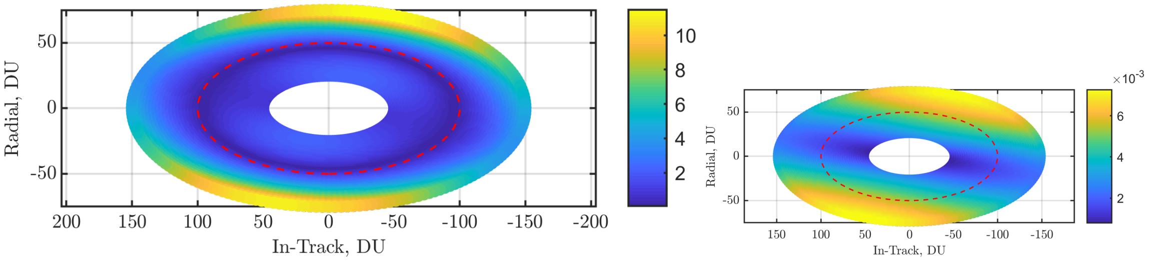
Unexpectedly, the controller also performs well for some initial states other than those on the specified NMC, as shown in Figure 31. For initial states on NMC's with smaller a_e values, the controller exhibits minimal terminal position error with marginally higher ΔV compared to optimal, however, performance drops off precipitously outside the initial NMC. This dichotomy may be due in part to the rudimentary network input normalization method employed. Since all position values are scaled by the largest position value of the initial NMC, this likely causes the larger position values outside this NMC to saturate the network nodes. Preliminary investigation suggests performance scales to some extent with the size of this initial NMC (see Figure 32), but further research is required. Improved generalized performance could also be derived from more sophisticated input normalization techniques. The geom-



(a) NC

(b) Optimal Control

Figure 30. Comparison of NC and Benchmark Optimal Controller

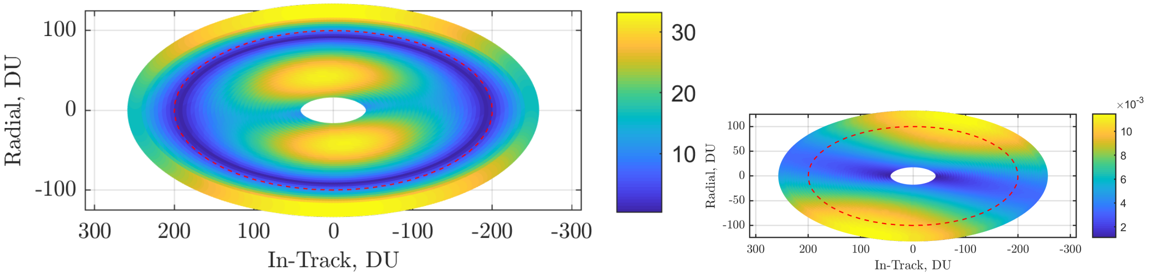


(a) Terminal Position Error, DU

(b) Total Control, DU/s

Figure 31. Generalization Performance: $a_e \in [50, 150]$
Dashed Line is $a_e = 100$ **Training Set**

etry of these plots may also inform the selection of a more efficient training set. The larger terminal position errors observed on the diagonal in Figure 31 suggest a higher concentration of training samples in these regions, instead of the linear spacing used, may improve efficiency and performance. Additionally, as identified in [52], the symmetry of the problem should be leveraged to decrease the training set size. Applying these techniques may reduce computation requirements while retaining equivalent controller performance, and vice versa.



(a) Terminal Position Error, DU

(b) Total Control, DU/s

Figure 32. Generalization Performance: $a_e \in [50, 250]$
Dashed Line is $a_e = 200$ **Training Set**

4.3 Problem C: Bounded Uncertainty

For Problem C, the following initial values are used:

Table 11. Problem C Initial ROE's

a_e	x_d	y_d	β	z_{max}	ψ
100	0	0	0	0	0

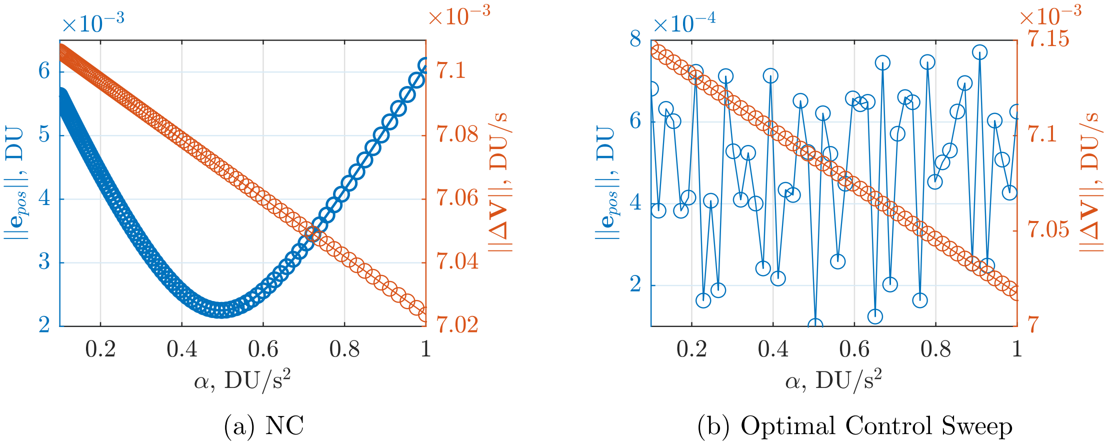
with a fixed final time of $t_f = 1/4\,\text{orbit}$. The origin is specified at an altitude of 35 786 km.

As in Problems A1 and A2, the position and velocity network inputs are scaled by $1 \cdot 10^{-2}$ and 100 respectively, with the uncertain $\hat{\alpha}$ input scaled by the inverse of the initial guess, $\hat{\alpha}_0$. The control outputs are scaled by $1 \cdot 10^{-2}$ and the output, $\hat{\alpha}$, by $\hat{\alpha}_0$. The initial guess is set to the middle of the α range at 0.5. The burn times are fixed at $\frac{1}{10}$ and $\frac{4}{10}$ of t_f.

The training set comprises 20 log-spaced values in the α range $[0.1, 1]$, with the number of samples in each batch, $s = 5$. For this problem, a logarithmic training set was found to converge faster and with greater consistency across the operating range than a linearly spaced set. The same terminal nonlinear inequality constraint as in Problem B, Eq. (4.10), is implemented with $\gamma = 1$ and $\epsilon_r = 1 \cdot 10^{-3}\,\text{DU}$. The cost function is the mean square of the total ΔV per sample, i.e.,

$$J = \frac{1}{s} \sum_{i=1}^{s} \sum_{j=1}^{n} \|\Delta \boldsymbol{V}_{j,i}\|_2^2 \tag{4.12}$$

where s is the number of samples in each batch, and n is the number of burns. As in Problem B, Figure 33 compares the NC results for several α values to an optimal benchmark solution. The benchmark was found by solving 50 individual static optimization problems using *fmincon* with 50 known α values. The total ΔV matches favorably with the benchmark optimal values, exhibiting acceptable, albeit higher terminal position error. The NC was simulated on 100 additional α values within the defined bounds to verify the neurocontroller's ability to generalize outside



(a) NC

(b) Optimal Control Sweep

Figure 33. Problem C Generalization and Performance Comparison

the training set with smooth results and no undesirable nonlinearities. Outside the bounds $[0.1, 1]$, position error increases as expected. Note that for this initial and final position, the perturbation actually assists the controller, hence the decrease in control with increasing α.

V. Conclusions and Recommendations

5.1 Review

In this research, neurocontrollers are developed for proximal spacecraft maneuvers. Three different thrust models are implemented: impulsive, finite, and continuous, in both open- and closed-loop control configurations. The controllers are applied to three common RPO maneuvers: intercept, rendezvous, and NMC transfers. The neurocontrollers are optimized numerically using both gradient-descent and genetic algorithms. The research combines aspects of traditional optimal control direct shooting methods with advances made in reinforcement learning. The methodology is implemented on three primary problems: Problem A applies the framework to the optimal control for three different sub-problems, while Problems B and C expand on these results by introducing a level of robustness. This research successfully demonstrates the initial application of neural networks as controllers to a variety of relative spacecraft maneuver problems; a few key insights are highlighted in the proceeding section, with a summary of future work discussed in § 5.3.

5.2 Insights

The *universal approximation* property of neural networks make them a flexible parameterization of optimal control problems. This is illustrated by the diversity of network inputs and outputs used throughout this research, from initial state vectors, to ROE's, ΔV, burn duration, and parameter estimates. Although not presented in this research, as discussed in § 4.3, one or more networks can be employed to approximate any number of desired mappings for a specific problem. Neural networks can be implemented in both open- and closed-loop configurations depending on the requirements of the problem and application. Significantly, this parameterization enables the generalized robustness demonstrated in Problems B and C.

For the problems solved in this research, surprisingly relatively small neural networks are required to achieve acceptable performance. All of the networks used required only one hidden layer and with the exception of Problem A3, only four hidden nodes. Although additional layers and nodes were explored, no benefit was observed, and the computational performance suffered due to the increase in design variables. As discussed in § 3.1.2, this parameterization does not lend itself to good initial guesses. Fortunately, performance and convergence stability were not observed to be overly sensitive to the initial guess. As well, despite their status as the *de facto* activation function for neural networks in current ML research, rectified linear units were not found to converge as well for these problems, compared to the hyperbolic tangent activations used.

The results in Problem B, specifically the unexpected performance outside the training range depicted in Figures 31 and 32, suggest a more efficient method exists to select training samples. Observing the higher terminal position error on the diagonal outside the training set, increasing the concentration of samples on this diagonal may result in more efficient optimization while retaining the same performance. Reference [52] leverages the symmetry of the problem to reduce the size of the required training set, and consequently the required computation time. A similar approach could be taken here. Alternatively, these results also suggest that more sophisticated input normalization techniques are warranted.

Ultimately, autonomy in the RPO domain requires robust and optimal controllers. Although applied to limited operating ranges, Problems B and C in particular demonstrate the potential of this methodology to develop neurocontrollers that fulfill this need. These operating ranges will of course need to be expanded and the networks correspondingly scaled to achieve the desired level of autonomy.

5.3 Future Research

5.3.1 Higher Fidelity Dynamics.

While the HCW equations of motion provide a convenient, computationally efficient model of relative satellite motion, they exhibit limited practical application. The optimization of neural networks is an inherently expensive computation due to the sheer quantity of parameters at play, and therefore HCW is a reasonable starting point but future work should examine more accurate, higher-order models like the ones identified in § 2.2.1 to include perturbed eccentric orbits and two-body motion. The direct shooting method employed in this research is amenable to most any dynamics model since no approximation of differential equations is made as in other OC techniques like collocation. The downside of course is the increased computation expense of such higher accuracy models. This may be alleviated in part by first solving the problem with more efficient, lower-accuracy dynamics and seeding the optimized network to be solved again using less efficient, but higher-accuracy dynamics. Additionally, the size of the requisite network may provide a metric for quantifying the complexity of the different dynamics models.

5.3.2 Larger, More Complex Training Sets.

For the problems solved in this research, neural networks are certainly not the best tool for the job, Problem A especially illustrates this; analytic solutions exist and other powerful optimal control techniques like pseuodspectral methods are far more efficient and accurate. Although Problems B and C demonstrate the potential of neural networks for autonomous spacecraft controllers, the specified operating ranges used significantly limit their utility. Undoubtedly larger, more complex operating ranges are required that will demand correspondingly larger, more complex networks. This will of course come at the cost of computation and problem tractability. Path constraints in particular often present a challenge to even the best numerical optimization tools. This research will need to be combined with advances being made in the field elsewhere in order to overcome such hurdles.

5.3.3 Comparison to Supervised Learning.

In [52] and [43], a supervised learning approach was taken to the problem of optimal spacecraft control trajectories. With SL, prior to optimizing the neural network, a large dataset must first be created, in the case of [43] over 300,000 trajectories. To generate these data sets, thousands of optimal control problems must first be solved typically using one of the methods outlined in § 2.3.2, e.g. collocation. While [43] employed a clever continuation technique to reduce the computation time, whereby nearby results are seeded to the optimizer for the next data point, this dataset generation step is potentially computationally prohibitive or intractable using traditional OC methods. However, once the dataset is created, the neural network can be optimized much more efficiently than with RL techniques, as in this research, using the increasingly robust and mature software tools built for the purpose of SL. This distinction is likely to become more pronounced as the complexity of the problem increases. Further research should examine the computational trade-off between the two approaches when applied to the same OC problem. It may be found that this trade-off is problem specific as well.

5.3.4 Multi-Player Differential Games.

Some of the initial inspiration for this research came from the headline-grabbing advances made by Google's DeepMind team with games like Go and later StarCraft.[35, 55] In particular, the goal is to apply the concepts and frameworks developed in this research to games of incomplete, asymmetric information for spacecraft in close proximity. This research should seek to develop and analyze the application of ML methods to the domain of optimal relative spacecraft control, building on the work of Stupik, Jagat, Cavalieri, Satak and Hurtado in differential game theory.[56–60]

5.4 Conclusion

This research successfully combines methods from reinforcement learning and traditional optimal control theory to develop spacecraft neurocontrollers. Both open- and closed-loop neurocontrollers are developed for three different thrust models to produce fuel-optimal trajectories for a variety of common spacecraft proximity maneuvers. Neural networks offer a flexible and powerful parameterization for complex optimal control problems in this domain. Undoubtedly wider and deeper networks will be needed as the designated operating ranges and difficulty of the specified problem scenarios increase. Future research should continue to expand the problem set, combining the robust qualities of bounded uncertainty and initial conditions as demonstrated separately in Problems B and C. Additionally, further network input and output representations should be explored to incorporate arbitrary targeting thereby expanding the valid operating range for a given controller. The computational performance trade-off of the two machine learning methods, supervised and reinforcement, should be assessed. The results shown in Problem B offer some insight into a more computationally efficient training set selection. The problems solved in the current research demonstrate the feasibility of using neural networks to create both optimal and robust controllers and present a path towards more autonomous spacecraft.

Bibliography

[1] Air Force Space Command, "Geosynchronous Space Situational Awareness Program," March 2017. `http://www.afspc.af.mil/`.

[2] Air Force Research Laboratory (AFRL), "Fact Sheet: Automated Navigation and Guidance Experiment for Local Space (ANGELS)," 2014. `https://www.kirtland.af.mil/`.

[3] Air Force Research Laboratory (AFRL), "ESPA Augmented Geosynchronous Laboratory Experiment (EAGLE)," 2018. `https://www.kirtland.af.mil/`.

[4] TRADOC G-2 and G. T. R. I. (GTRI), "Robotics, Artificial Intelligence & Autonomy: Visioning Multi-Domain Warfare in 2030-2050," Tech. Rep. March, 2017.

[5] D. A. Vallado and W. D. McClain, *Fundamentals of astrodynamics and applications.* Microcosm Press, 3 ed., 2007.

[6] M. Willis, T. A. Lovell, and S. D. Amico, "Second-Order Analytical Solution for Relative Motion on Arbitrarily Eccentric Orbits," in *29th AIAA/AAS Sp. Flight Mech. Meet.*, pp. 1–23, 2019.

[7] J. Sullivan, S. Grimberg, and S. D'Amico, "Comprehensive Survey and Assessment of Spacecraft Relative Motion Dynamics Models," *J. Guid. Control. Dyn.*, vol. 40, no. 8, pp. 1837–1859, 2017.

[8] W. H. Clohessy and R. S. Wiltshire, "Terminal Guidance System for Satellite Rendezvous," *J. Aerosp. Sci.*, vol. 27, no. 9, pp. 653–658, 1960.

[9] M. L. Anthony and F. T. Sasaki, "The Rendezvous Problem for Nearly Circular Orbits," in *AIAA Aerosp. Sci. Meet.*, no. 65-32, (New York), 1965.

[10] B. A. Newman, A. J. Sinclair, A. Lovell, and A. Perez, "Comparison of Nonlinear Analytical Solutions for Relative Orbital Motion," in *AIAA/AAS Astrodyn. Spec. Conf.*, pp. 1–29, AIAA, 2014.

[11] E. A. Butcher E. Burnett and T. A. Lovell, "Comparison of Relative Orbital Motion Perturbation Solutions in Cartesian and Spherical Coordinates," *27th AAS/AIAA Sp. Flight Mech. Meet.*, vol. San Antonio, February 2017.

[12] P. R. Hempel and J. F. A. Tschauner, "Minimum-Fuel Rendezvous Techniques," *J. Spacecr. Rockets*, vol. 2, no. 5, pp. 802–804, 1965.

[13] T. E. Carter, "State Transition Matrices for Terminal Rendezvous Studies: Brief Survey and New Example," *J. Guid. Control. Dyn.*, vol. 21, no. 1, pp. 148–155, 1998.

[14] K. Yamanaka and F. Ankersen, "New State Transition Matrix for Relative Motion on an Arbitrary Elliptical Orbit," *J. Guid. Control. Dyn.*, vol. 25, no. 1, pp. 60–66, 2002.

[15] S. A. Schweighart and R. J. Sedwick, "High-Fidelity Linearized J Model for Satellite Formation Flight," *J. Guid. Control. Dyn.*, vol. 25, no. 6, pp. 1073–1080, 2002.

[16] D.-W. Gim and K. T. Alfriend, "State Transition Matrix of Relative Motion for the Perturbed Noncircular Reference Orbit," *J. Guid. Control. Dyn.*, vol. 26, no. 6, pp. 956–971, 2003.

[17] T. A. Lovell and S. Tragesser, "Guidance for Relative Motion of Low Earth Orbit Spacecraft Based on Relative Orbit Elements," in *AIAA/AAS Astrodyn. Spec. Conf. Exhib.*, pp. 1–16, 2004.

[18] T. A. Lovell and D. L. Brown, "Impulsive-Hover Satellite Trajectory Design for Rendezvous and Proximity Operation Missions," in *AAS/AIAA Sp. Flight Mech. Meet.*, pp. 1–16, 2007.

[19] K. Ogata, *Modern Control Engineering.* Prentice Hall, 5 ed., 2010.

[20] A. E. Bryson, "Optimal Control–1950 to 1985," *IEEE Control Syst.*, pp. 1–88, 1996.

[21] F. L. Lewis, V. L. Syrmos, and D. L. Vrabie, *Optimal Control.* Wiley, 2012.

[22] J. T. Betts, "Survey of Numerical Methods for Trajectory Optimization," *J. Guid. Control. Dyn.*, vol. 21, no. 2, pp. 193–207, 1998.

[23] M. Kelly, "An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation," *SIAM Rev.*, vol. 59, no. 4, pp. 849–904, 2017.

[24] A. V. Rao, "A Primer on Pseudospectral Methods for Solving Optimal Control Problems," 2012.

[25] J. Slotine and W. Li, *Applied Nonlinear Control.* Englewood Cliffs, NJ: Prentice-Hall, 1991.

[26] J. S. Arora, *Introduction to Optimum Design.* Academic Press, 4 ed., 2017.

[27] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016.

[28] A. L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers," *IBM J.*, 1959.

[29] K. Hao, "We analyzed 16,625 papers to figure out where AI is headed next," *MIT Technol. Rev.*, January 2019.

[30] Cognub, "Cognitive Computing and Machine Learning." http://www.cognub.com/.

[31] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "OpenAI Gym," *arXiv*, pp. 1–4, 2016.

[32] A. Moore, "Efficient memory-based learning for robot control," Tech. Rep. UCAM-CL-TR-209, University of Camridge, Cambridge, 1990.

[33] R. S. Sutton and A. G. Barto, *Reinforcement Learning - An Introduction*. Adaptive Computation and Machine Learning, MIT Press, 1998.

[34] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," *arXiv*, pp. 1–9, 2013.

[35] H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-learning," *Assoc. Adv. Artif. Intell.*, 2016.

[36] G. Tesauro, "TD-gammon, a self-teaching backgammon program, achieves master-level play," *Neural Comput.*, vol. 6, pp. 215–219, 1994.

[37] K. G. Vamvoudakis and F. L. Lewis, "Online actor-critic algorithm to solve the continuous-time infinite horizon optimal control problem," *Automatica*, vol. 46, no. 5, pp. 878–888, 2010.

[38] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal and Autonomous Control Using Reinforcement Learning: A Survey," *IEEE Trans. Neural Networks Learn. Syst.*, 2018.

[39] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution Strategies as a Scalable Alternative to Reinforcement Learning," *arXiv*, pp. 1–13, 2017.

[40] S. Jadon, "Introduction to Different Activation Functions for Deep Learning," March 2018. https://medium.com/.

[41] G. Cybenko, "Degree of approximation by superpositions of a sigmoidal function," *Approx. Theory its Appl.*, vol. 9, no. 3, pp. 17–28, 1989.

[42] I. Carnelli, B. Dachwald, and M. Vasile, "Evolutionary Neurocontrol: A Novel Method for Low-Thrust Gravity-Assist Trajectory Optimization," *J. Guid. Control. Dyn.*, vol. 32, no. 2, pp. 616–625, 2009.

[43] D. Izzo, C. I. Sprague, and D. Tailor, "Machine learning and evolutionary techniques in interplanetary trajectory design," *arXiv*, pp. 1–20, 2018.

[44] C. Sánchez-Sánchez and D. Izzo, "Real-time optimal control via Deep Neural Networks: study on landing problems," *J. Guid. Control. Dyn.*, pp. 1–14, 2016.

[45] A. Mereta, D. Izzo, and A. Wittig, “Machine learning of optimal low-thrust transfers between near-earth objects,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10334 LNCS, pp. 543–553, 2017.

[46] T. Vinko, D. Izzo, and C. Bombardelli, “Benchmarking Different Global Optimisation Techniques for Preliminary Space Trajectory Design,” *58th Int. Astronaut. Congr.*, 2007.

[47] J. F. Horn, E. M. Schmidt, B. R. Geiger, and M. P. DeAngelo, “Neural Network-Based Trajectory Optimization for Unmanned Aerial Vehicles,” *J. Guid. Control. Dyn.*, vol. 35, no. 2, pp. 548–562, 2012.

[48] D. L. Yang, B. Xu, and L. Zhang, “Optimal low-thrust spiral trajectories using Lyapunov-based guidance,” *Acta Astronaut.*, vol. 126, pp. 275–285, 2015.

[49] D. C. Dracopoulos and A. J. Jones, “Adaptive neuro-genetic control of chaos applied to the attitude control problem,” *Neural Comput. Appl.*, vol. 6, no. 2, pp. 102–115, 1997.

[50] D. C. Dracopoulos, “Evolutionary Control of a Satellite,” in *Genet. Program.*, pp. 77–81, Stanford University, 1997.

[51] D. C. Dracopoulos, *Evolutionary learning algorithms for neural adaptive control.* Springer, 1997.

[52] E. A. Youmans and F. H. Lutze, “Neural Network Control of Space Vehicle Intercept and Rendezvous Maneuvers,” *J. Guid. Control. Dyn.*, vol. 21, no. 1, pp. 116–121, 1998.

[53] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” *arXiv*, 2015.

[54] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control: An Introduction.* Prentice-Hall, 1991.

[55] O. Vinyals, T. Ewalds, S. Bartunov, *et al.*, “StarCraft II,” *arXiv*, 2017.

[56] J. M. Stupik, *Optimal Pursuit/Evasion Spacecraft Trajectories in the Hill Reference Frame.* Masters thesis, University of Illinois at Urbana-Champaign, 2013.

[57] A. Jagat and A. J. Sinclair, “Optimization Of Spacecraft Pursuit-Evasion Game Trajectories In The Euler-Hill Reference Frame,” in *AIAA/AAS Astrodyn. Spec. Conf.*, no. August, pp. 1–20, 2014.

[58] T. D. Woodbury and J. E. Hurtado, “Adaptive play via estimation in uncertain nonzero-sum orbital pursuit evasion games,” *AIAA Sp. Astronaut. Forum Expo.*, pp. 1–15, 2017.

[59] K. A. Cavalieri, N. Satak, and J. E. Hurtado, "Incomplete Information Pursuit-Evasion Games with Uncertain Relative Dynamics," in *AIAA Guid. Navig. Control Conf.*, no. January, pp. 1–8, 2014.

[60] N. Satak, *Behavior learning in differential games and reorientation maneuvers.* PhD thesis, Texas A&M University, 2013.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 03/21/2019	**2. REPORT TYPE** Master's Thesis	**3. DATES COVERED** *(From - To)* Sep 2017 - Mar 2019

4. TITLE AND SUBTITLE
Optimal and Robust Neural Network Controllers for Proximal Spacecraft Maneuvers

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
George, Brandon, C, Capt

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way
Wright-Patterson AFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT-ENY-MS-19-M-215

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Labs, Space Vehicles Directorate
Dr. Alan Lovell, Dr. Andrew Sinclair
3550 Aberdeen Avenue SE
Kirtland AFB, NM 87117

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RV

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution Statement A. Approved for Public Release; Distribution is Unlimited

13. SUPPLEMENTARY NOTES
This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

In this research, reinforcement learning techniques are combined with traditional direct shooting methods to generate optimal proximal spacecraft maneuvers. Open- and closed-loop controllers, parameterized by neural networks, are developed for terminally constrained, fuel-optimal relative motion trajectories using three different thrust models. Neurocontroller performance robustness to parametric uncertainty and bounded initial conditions is assessed. This research demonstrates that neurocontrollers offer a flexible and robust alternative approach to the solution of complex controls problems in the space domain and present a promising path forward to more capable, autonomous spacecraft.

15. SUBJECT TERMS
relative satellite motion, neural networks, optimal control, reinforcement learning, direct shooting

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Maj Joshuah Hess, AFIT/ENY
					19b. TELEPHONE NUMBER *(Include area code)*
U	U	U	UU	90	(937) 255-3636 x4713